Ramesh Manian | Mini Project - Part 2 | rmanian@stanford.edu
Uzair Mansuri | umansuri@stanford.edu

1 Abstract

In part 2 of our mini-project, we evaluated the performance of our winning regression model of Gross revenue on a held-out test set. Further, we fitted linear models to the train and test sets, from which we employed inference techniques to analyze the significance and spread of our coefficient estimates.

2 Prediction on the test set

Before evaluating the test error from our regression models, we applied the same scaling and variable transformations from the training set to the test set, in order to make sure we are working with variables that are transformed in a similar fashion as the training set we trained our model on, to reduce bias.

Regression Model

For our winning regression model, we used '`finalModel`' with covariates selected based on different modeling approaches we implemented in Part 1 of the project. We applied `finalModel` on the held out test set and compared the results with our estimate of the generalization error using cross-validation.

	winning Model
Training Error	85,532,912
CV Error (K=10)	89,541,120
Test CV Error	99,580,290

Table 1: Error comparison for Winning Model

Table 1 above provides a summary of regression results for our Winning model. In order to estimate the generalization error, we used cross-validation with 10 folds ($K = 10$). Primarily, there are two main reasons why the cross validation error can be a biased estimate of the true generalization error. First one is that we used the same number of folds ($K = 10$) to estimate the test error, for each of our models and down-selected the best model according to lowest CV error. Thus, the estimate of generalization error by cross validation for our chosen `finalModel` is an under estimate of the true test error as there is selection bias.

The second reason is that when we used cross-validation with ($K = 10$) folds, we trained our model on 9 out of 10 training data and evaluate on the remaining training data. The resulting estimate is the mean of 10 errors by carrying out this procedure. However, when training our `finalModel`, we train using entire training data and get the test error listed in Table 1, by applying our model on the test set, this leads to an over estimation of test error as we do not train `finalModel` on the entire set of training data.

Since we have a large dataset, we can see the impact of second effect is minimal as compared to first one, from the results shown in Table 1, which shows cross validation error to be an underestimate of Test error. Also, the training error, as expected is lower than both CV error and test error as the model is designed such that it minimizes the training error.

3 Inference

For inference, we have used our winning linear regression model from Part 1 with covariates selected using feature selection, transformation methods applied and elimination of highly correlated variables from the model.

3.1 Statistical significance

We make the following assumptions on the population model namely: that it is linear, has i.i.d errors with mean zero and constant variance (Homoscedastic). Based on the above assumptions, statistical significance in statistics is described by the term 'P Value', which is defined as the probability of observing a test statistic that is as extreme as the one observed, if the null hypothesis were true. We have $\beta_j = 0$ as our null hypothesis.

So, if we take for example, the coefficient value for the `Runtime` covariate, we can see that the t value, which is a ratio of the estimate to the standard error of the regression output for that specific covariate, is -2.641 as shown in Table 2. This says if the true value of the coefficient were zero, then the probability of observing the specific statistic value as extreme as the one we got, $|t| = 2.641$, is just over nine thousandth of one percent. On the basis of this strength, we will reject the null hypothesis.

Significance can be at multiple levels, 5%, 1% or 0.1%. The 5% level is generally referred to as "Statistically Significant" and for our project, we have selected that as the threshold. From the regression output, we see that only 6 out of 30 covariates are not statisti-

Covariate	Estimate	$\widehat{\text{SE}}$	t value	Pr($>\|t\|$)
Intercept	-2.23e+10	1.05e+10	-2.118	0.034
Runtime	-7.02e+07	2.65e+07	-2.641	0.008
imdbVotes	7.42e+02	5.17e+01	14.352	<2e-16
Budget	3.62e+01	7.05e+00	5.143	2.86e-07
tomatoUserRating	1.28e+10	5.94e+09	2.155	0.031
tomatoReviews	-1.907e+05	1.805e+05	-1.056	0.291

Table 2: Linear Regression results - Training Data

cally significant; covariates are marked with asterisk if they are significant. The regression output tells us that all covariates involving `tomatoFresh` are not significant. From a common sense perspective, it is slightly surprising that `tomatoReviews`, which is the total number of tomato reviews by critics was not reported as significant.

We believe that the results from the regression output are consistent with our experience with the data. For instance, it is very reasonable that a movie goer will look at the `imdbRating` or a `tomatoUserRating` or look at how many votes the movie garnered (`imbdVotes`) prior to investing going to watch a movie. Therefore, it is reasonable to expect that these covariates will influence the revenue that the movie makes. Thus we reject the hypothesis that the coefficient is zero.

As second part of our analysis, we fitted the same model on the test data to evaluate the effectiveness of our winning model to a held out test and the regression output was quite surprising. However, the RMSE for test was approximately $79.41M vs $85.53M we got for training. The R-squared remained essentially the same. The model fitted with test data was also different from the point of covariates' significance. Only 16 covariates were deemed significant as opposed to 24 covariates in the model fitted with training data.

We also found that `tomatoReviews`, which was not significant in our earlier model with training data, is now significant at the 95% level. Training `imdbRating` was surprisingly not significant in test ($p = 0.25124$) while it was significant at 1% ($p = 0.00324$).

The results from our model evaluation on test data is shown in Table 3 below:

Covariate	Estimate	$\widehat{\text{SE}}$	t value	Pr($>\|t\|$)
Intercept	7.147e+09	2.806e+10	0.255	0.799
Runtime	1.557e+06	7.182e+07	0.022	0.982
imdbVotes	8.527e+02	1.183e+02	7.208	1.34e-12
Budget	2.445e+01	1.494e+01	1.636	0.102
tomatoUserRating	-5.013e+09	1.632e+10	-0.307	0.758
tomatoReviews	3.484e+04	3.275e+05	0.106	0.915

Table 3: Linear Regression results - Test Data

Based on our results from the test data, we found that the p-values and t statistic for the estimates have changed when compared to test data, specifically highly significant covariates with p-values $< 5\%$ such as `Runtime` and `Budget` as an example, are not signficant anymore based on their p-values $> 5\%$. One way this can be explained is that we have much less data in our test set ($n = 816$), compared to ($n = 3277$) for the training set, in order to build our estimates that results in less evidence whether a coefficient is zero. Another reason why we are seeing differences in significance of our results when we apply our model to the test set is because of *post-selection inference* as described in further detail in Section 3.3.1.

As discussed above, our `finalModel` was a subset of the original covariates plus several transformed covariates as well as some interaction terms between the covariates. When we computed a kitchen sink model, which included all the covariates from our dataset and all possible interaction terms, the model had two covariates that were significant that were also significant in our `finalModel`:`Runtime` and `tomatoUserRating`. The `kithcenSinkModel` also had covariates such as `imdbVotes` and `Budget` as non-significant whereas it was the opposite in the final chosen model.

Further, the coefficient value `Runtime` was significant ($p = 0.0000454$) at 1% level with a coefficient of $5.279e + 07$ in the kitchenSinkModel. This is very different from what we have in the final model, which

had a coefficient of $-7.023e+07$ which was also significant ($p = 0.008311$). It is hard to conclude anything meaningful from this alone, except that this is an important covariate that must be taken into consideration in the model. Additionally, The `kitchenSinkModel` was rank-deficient and several covariates as undefined because of singularities.

3.2 Bootstrap

We used bootstrap technique (case re-sampling) to have an estimate of sampling distribution on the estimates of the coefficients for the linear regression model. Specifically, we are looking to compute confidence intervals for our regression coefficients.

By bootsrapping, we draw $n = 1000$ samples randomly from the training set with replacement and fit a linear regression model on each of this bootstrap sample. As a result, we compute a bootstrap distribution, standard error and confidence intervals for each coefficient, which are used to compare against values given by our linear regression model.

We used Quantile interval method to compute the following results from Bootstrap. Note that we have only selected few covariates for our discussion, which are representative of results.

Covariate	Estimate	$\hat{SE}$	CI at 95%
Intercept	-2.23e+10	1.48e+10	[-6E+10, 3E+09]
Runtime	-7.02e+07	4.09e+07	[-2E+08, -2E+06]
imdbVotes	7.42e+02	1.01e+02	[537.5, 953]
Budget	3.62e+01	1.77e+01	[3.45, 70.85]
tomatoUserRating	1.28e+10	8.42e+09	[-6E+09, 3E+10]
Runtime:Budget	7.31e-02	4.06e-02	[-0.0001, 0.1565]

Table 4: Bootstrap results

Table 4 provides a summary of our Bootstrap results. In order to compare the results from bootstrap, we summarize the results for similar covariates from our linear regression model.

Covariate	Estimate	$\hat{SE}$	CI at 95%
Intercept	-2.23e+10	1.05e+10	[-4E+10, -2E+09]
Runtime	-7.02e+07	2.65e+07	[-1E+08, -2E+07]
imdbVotes	7.42e+02	5.17e+01	[640.9, 843.74]
Budget	3.62e+01	7.05e+00	[22.44, 50.09]
tomatoUserRating	1.28e+10	5.94e+09	[1.2E+09, 2E+10]
Runtime:Budget	7.31e-02	1.50e-02	[0.044, 0.103]

Table 5: Linear Regression results

As seen from Tables 4 and 5, we can see that the standard errors computed by bootstrap are higher when compared to the ones computed by standard OLS. Because the bootstrap method of computing standard errors does not rely on the assumption that errors are normally distributed, it makes sense that the bootstrapped standard errors would be larger. Additionally, the confidence intervals for intercept has also shifted while running bootstrap, when other coefficients change as seen in Table 4. However, there is still a significant overlap between CI of intercept between two models in our analysis.

3.3 Potential problems with Linear model

3.3.1 Post-selection Inference

From our analysis, we've noticed that the p-values of coefficient estimates on the test set were higher when compared to the ones in the training set. These differences can be explained by post-selection inference, as we have selected our covariates and winning model based on the training data, as a result of which we've optimized their significance and biased our selection of p-values. When we applied our winning model on the test set, the significance for many covariates changed and reasonably, p-value also increased. Thus,

we need to be careful about how we infer *'statistical significance'* for our `finalModel`, as it has been trained using training set and it's significance changes when applied to held out test set.

3.3.2 Multiple hypothesis testing

As we've discussed during lectures, running multiple hypothesis tests simultaneously on coefficients is also a risk to our analysis. For all covariates selected in our `finalModel`, we always have a risk that we might've ended up selecting non-significant covariates. By deciding on 5% cutoff for p-value, we accept $\approx 5\%$ false positives. To ensure that the probability of declaring even one false positive is not greater than 5%, we can use multiple testing corrections such as the Bonferroni and Benjamini-Hochberg correction.

Because we are testing the significance of multiple variables, if we considered all features with p-values < 0.05 to be significant, we would expect to reject 5% of null hypotheses even when they are correct. After applying Bonferroni correction, we can see from Figure 1 that many covariates show a value of 1, indicating they are above the cutoff of $0.05/p$ on the p-value, which are deemed as not significant.However, we know that Bonferroni is very conservative as the number of coefficients increases. To overcome this limitation of Bonferroni, we decided to apply Benjamini-Hochberg correction where we found that p-values remained significant for variables selected via recursive feature elimination.

	OLS	Benjamini-Hochberg	Bonferroni
(Intercept)	0.0342823	0.0442814	1.0000000
Runtime	0.0083106	0.0151547	0.2576291
imdbRating	0.0919365	0.1140013	1.0000000
imdbVotes	0.0000000	0.0000000	0.0000000
tomatoReviews	0.2910372	0.3111087	1.0000000
tomatoFresh	0.2799384	0.3099318	1.0000000
tomatoUserRating	0.0312091	0.0420644	0.9674815
Budget	0.0000003	0.0000011	0.0000089
imdbVotes_bin	0.4674563	0.4674563	1.0000000
Budget_bin	0.0000282	0.0000795	0.0008748
Runtime_log	0.0270714	0.0382326	0.8392119
imdbRating_P2	0.3105652	0.3209173	1.0000000
imdbVotes_P2	0.0000000	0.0000000	0.0000000
tomatoUserRating_P2	0.0271328	0.0382326	0.8411177
Runtime:tomatoFresh	0.0080511	0.0151547	0.2495855
Runtime:tomatoUserRating	0.0051794	0.0107041	0.1605616
Runtime:Budget	0.0000013	0.0000039	0.0000393
Runtime:Budget_bin	0.0000359	0.0000928	0.0011142
Runtime:imdbVotes_P2	0.0000000	0.0000000	0.0000000
Runtime:tomatoUserRating_P2	0.0031550	0.0069861	0.0978051
imdbRating:tomatoFresh	0.2351552	0.2699930	1.0000000
imdbVotes:tomatoReviews	0.0003706	0.0008838	0.0114895
imdbVotes:tomatoFresh	0.1701276	0.2028445	1.0000000
imdbVotes:Budget	0.0000000	0.0000000	0.0000000
tomatoReviews:imdbRating_P2	0.0000000	0.0000000	0.0000000
tomatoReviews:tomatoUserRating_P2	0.0000000	0.0000000	0.0000000
tomatoUserRating:Runtime_log	0.0227631	0.0352828	0.7056558
Budget:imdbVotes_bin	0.0154942	0.0266844	0.4803199
Budget:Runtime_log	0.0000006	0.0000019	0.0000174
Runtime_log:imdbVotes_P2	0.0000000	0.0000000	0.0000000
Runtime_log:tomatoUserRating_P2	0.0188751	0.0307962	0.5851272

Figure 1: P value comparison

Figure 1 shows comparison between P-values between standard OLS model and both correction methods (Benjamini-Hochberg and Bonferroni).

4 Discussion

In this section, we would like to discuss about practicality of using our model in real world to predict Gross revenues for Movies, how the model will hold up over time and what are some of the caveats or points a person inferring our model needs to be aware of. Finally, we would like to have some dialogue about what we would've done differently knowing what we know now, before tackling the same dataset again.

4.1 Practical use of our model

The methodology that we used, notwithstanding our conclusion on what we would do differently if we were to do it again, are all very practical and our models are realistic and can be applied practically. We could envision using this model to do some hypothesis testing with some refinements discussed. We could try and gain causal inferential knowledge about profitability as a causal relationship between budget and revenue. We could also use the methods and models shown here to identify marketing strategies.

Models discussed here are appropriate for both prediction and inference as you see in the discussions above. The analysts using this model have to take a lot of care to ameliorate the data issues discussed above, including imputation of data. The data that we used carry information spans nearly a century and can thus be stale from the perspective of ground truth. For instance, a movie made in the 70's may be irrelevant for today's generation and the data associated with it will be meaningless from both prediction and inference. So, we would suggest new analysts to pick data from a period such as post year 2000. While the model strategies should work well in the future, it is important to keep the data fresh and relevant and therefore, refresh the models periodically when enough new data are acquired.

We also need to refresh the model as the interaction between covariates and the relationship between the outcome and covariate can change. For example, Western movies may no longer be in vogue in 2021 and therefore, training data will have to adjust for that ground truth.

4.2 Model hold up over time

The Gross revenues for a movie depends on a lot of factors, and since the market landscape is very competitive and always changing, we expect that our model will need to be refitted quite often. It should be refitted when the training data fails to account for price fluctuations pertaining to movie industry. For instance, if the industry underwent a sudden shock, perfect exam-

ple is ongoing COVID-19 situation where the overall revenues for movies are at historical lower levels, our model, which is trained on historical data, would not generalize well. Similarly, if new trend appears that can disrupt the market, our models would likely systematically overestimate Gross revenues, resulting in increased error. Our models should be refitted immediately after such extreme events so that the training data better resembles the actual test data.

4.3 Collinearity

We believe that there were several covariates that were indeed collinear. The `kitchenSinkModel` was shown to have twelve covariates that were collinear as indicated by regression output message, *"Coefficients: (12 not defined because of singularities)*. In examining our corrplot and data more closely, we also found that several covariates were very highly correlated, `tomatoUserRating` and `tomatoUserMeter` for instance with correlation of greater than 0.8, which also indicated that collinearity was affecting our results. We proceeded to manually remove these variables for our final model.

4.4 Data collection process

As discussed during Part 1 of our project, '`movies merged`' dataset used in our Project was compiled by Dr. Guy Lebanon, an ex-professor at Georgia Tech. Even though the data collection process appeared satisfactory to us, we would've collected additional data/-covariates, that might've helped us in understanding our population model even better. As an example, covariates on advertising data to assess the impact of movie advertising on revenue and covariate/data on piracy that would hurt movie revenues could be collected and included in our model for better prediction. Also, data on the impact of COVID-19 on the global film industry would've been very interesting to know about, which can increase overall prediction robustness of similar models to market shock events.

4.5 Points to be aware of

Someone looking at our analysis should be aware that our dataset contains movie revenues starting from year 1888 to as recent as 2018, and the model is not adjusted for inflation, nor does it account of changing pipeline of evolving people interest or technological disruption over time which means our model would likely not generalize well across entire time period.

Because our features include some collinearity, the coefficients on some features, particularly the two describing movie ratings (`imdbRating` and `tomatoUserRating`), might not fully reflect the relationship between Gross revenue and various features. As discussed in section 3.3.1 above, we also performed feature selection before fitting a model, which means we might be overstating the correlation between all the covariates and Gross revenue.

4.6 What would we do differently?

There are several things we would have done differently if we knew then what we know at the end of the class. We believe that we would have thought about the model from an end goal perspective; what did we want to really achieve? If as a studio, our main goal was profit, we would have liked to have inferred the influence of the causal relationship between Budget and Revenue covariates.

We would also have spent more time in covariate selection and additional regression techniques like Lasso and Ridge, instead of superficial handling of these methods. Additionally, we would have a better imputation of missing data than the route we took, which was to remove all rows with even one column of NA value. This would have given us almost 30K observations as opposed to the less than 4500 observations that we used to train our model. We would also have been more diverse and chosen more classification outcome variables, than just the hit or miss we did earlier.

5 References

- **Dataset**: `https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged`
- **MS&E 226 Lecture Notes**: `https://web.stanford.edu/class/msande226/l_notes.html`
- **R for Data Science**: `https://r4ds.had.co.nz/`
- **Cookbook for R**: `http://www.cookbook-r.com/`
- **Cross Validated**: `https://stats.stackexchange.com/`

MS&E 226: Project Part-2 (APPENDIX)

Ramesh Manian (rmanian@stanford.edu) & Uzair Mansuri (umansuri@stanford.edu)

11/20/2020

Project group members: *Ramesh Manian, Uzair Mansuri.*

Set seed for repeatability

```
set.seed(3945827)
```

Load the dataset and explore

```
# Extracting Data & creating Dataframe

df_movies <- load("C:/Users/uzair/OneDrive/MS&E 226 - Project 1/netflix/movies_merged")
df = movies_merged

# Gathering information on explanatory variables for 40,789
# observations
cat("Column names:", end = "\n", file = "")
```

```
## Column names:
```

```
colnames(df)
```

```
##  [1] "Title"             "Year"              "Rated"
##  [4] "Released"          "Runtime"           "Genre"
##  [7] "Director"          "Writer"            "Actors"
## [10] "Plot"              "Language"          "Country"
## [13] "Awards"            "Poster"            "Metascore"
## [16] "imdbRating"        "imdbVotes"         "imdbID"
## [19] "Type"              "tomatoMeter"       "tomatoImage"
## [22] "tomatoRating"      "tomatoReviews"     "tomatoFresh"
## [25] "tomatoRotten"      "tomatoConsensus"   "tomatoUserMeter"
## [28] "tomatoUserRating"  "tomatoUserReviews" "tomatoURL"
## [31] "DVD"               "BoxOffice"         "Production"
## [34] "Website"           "Response"          "Budget"
## [37] "Domestic_Gross"    "Gross"             "Date"
```

Cleaning up data

Remove non-movie rows

```
# Remove all rows from df that do not correspond to movies
df2 <- df[df$Type == "movie", ]
dim(df2)
```

```
## [1] 40000    39
```

```
# save df2 as df
df <- df2

# rem_cols <- c(tomatoConsensus, Plot, Poster, Website,
# imdbID, tomoatoImage, tomatoConsensus, BoxOffice,
# tomatoURL, DVD, Response, Language, Country)
```

Drop rows with missing `Gross` value

our goal is to model `Gross` revenue against other variables, therefore, rows that have missing `Gross` values are not useful to us.

```
# Remove rows with missing Gross value subset rows with valid
# gross values
df_gross_val = subset(df, !is.na(df$Gross))
df = df_gross_val
cat("Movies only dataset with valid Gross value has", dim(df)[1],
    "samples and", dim(df)[2], "features", "\n")
```

```
## Movies only dataset with valid Gross value has 4558 samples and 39 features
```

Process `Runtime` column

The variable `Runtime` represents the length of the title as a string. Let us convert it to a numeric value (in minutes) and replace `Runtime` with the new numeric column.

```
# Replace df$Runtime with a numeric column containing the
# runtime in minutes look for three time patterns in run_time
# 'xx h yy min' or 'xx h' or 'xx min'
runtime_pattern = c("(\\d+)\\sh\\s(\\d+)\\smin", "(\\d+)\\sh",
    "(\\d+)\\smin")
normalize_NA_convert_to_min <- function(s) {
    # convert N/A to NA
    if (s == "N/A" || s == "n/a" || s == "N/a" || s == "n/A")
        return(NA)
    # go through the possible ways time can be shown
    for (i in 1:length(runtime_pattern)) {
        val = str_match(s, runtime_pattern[i])
        if (!is.na(val[1])) {
            if (i == 1)
```

```
                return(as.numeric(val[2]) * 60 + as.numeric(val[3])) else if (i == 2)
                return(as.numeric(val[2]) * 60) else if (i == 3)
                return(as.numeric(val[2]))
        }
    }
}
df$Runtime = sapply(df$Runtime, normalize_NA_convert_to_min)
```

Let us extract valid observations; ones with budget information in our case

```
# Remove rows that have NA values in runtime and budget
# columns
df_rt_budget <- subset(df, !is.na(df$Runtime) & !is.na(df$Budget))
# 4520 observations are there with just Runtime and Budget
# being non-NA
cat("Number of rows to be removed with non NA values in Runtime and budget :",
    nrow(df_rt_budget))
```

```
## Number of rows to be removed with non NA values in Runtime and budget : 4520
```

Eliminate mismatched rows

Compare the `Released` column (string representation of release date) with either `Year` or `Date` (numeric representation of the year) to find mismatches.

```
# Remove mismatched rows Remove all rows with
# Year/Date/release mismatch
correct_year_mismatch = function(dft) {
    dft["YearRel"] = as.numeric(format(as.Date(dft$Released,
        format = "%Y-%m-%d"), "%Y"))
    c1 = (is.na(dft$Date) | (dft$Year == dft$Date) | (!is.na(dft$YearRel) &&
        dft$YearRel == dft$Date))
    dft = dft[c1, ]
    # delete temporary column or numeric representation of
    # Released
    dft = subset(dft, select = -YearRel)
    return(dft)
}

df_non_mismatched_year = correct_year_mismatch(df)
new_cnt = nrow(df_non_mismatched_year)
# compute percentage of mis-matched rows removed
df = df_non_mismatched_year
cat("Non-mismatched dataset has", dim(df)[1], "samples and",
    dim(df)[2], "features", "\n")
```

```
## Non-mismatched dataset has 4558 samples and 39 features
```

Drop `Domestic_Gross` column

`Domestic_Gross` is basically the amount of revenue a movie earned within the US. Quite likely, it is very highly correlated with `Gross` and is in fact equal to it for movies that were not released globally. Hence, we will remove it for modeling purposes.

```
# Exclude the 'Domestic_Gross' column
df_gross_removed = subset(df, select = -Domestic_Gross)
df = df_gross_removed
cat("Domestic_Gross removed dataset has", dim(df)[1], "samples and",
    dim(df)[2], "features", "\n")
```

```
## Domestic_Gross removed dataset has 4558 samples and 38 features
```

Final preprocessed dataset

Let us look at the dimensions of the preprocessed dataset that we will be using for modeling and evaluation. Let us also print all the final covariates list.

```
# Print the dimensions of the final pre-processed dataset and
# column names
cat("Pre-processed dataset has", dim(df)[1], "samples and", dim(df)[2],
    "features. Features are printed below.", "\n")
```

```
## Pre-processed dataset has 4558 samples and 38 features. Features are printed below.
```

```
pre_processed_df = df
print(sort(colnames(df)))
```

```
##  [1] "Actors"            "Awards"            "BoxOffice"
##  [4] "Budget"            "Country"           "Date"
##  [7] "Director"          "DVD"               "Genre"
## [10] "Gross"             "imdbID"            "imdbRating"
## [13] "imdbVotes"         "Language"          "Metascore"
## [16] "Plot"              "Poster"            "Production"
## [19] "Rated"             "Released"          "Response"
## [22] "Runtime"           "Title"             "tomatoConsensus"
## [25] "tomatoFresh"       "tomatoImage"       "tomatoMeter"
## [28] "tomatoRating"      "tomatoReviews"     "tomatoRotten"
## [31] "tomatoURL"         "tomatoUserMeter"   "tomatoUserRating"
## [34] "tomatoUserReviews" "Type"              "Website"
## [37] "Writer"            "Year"
```

Model Evaluation

Numeric variables

We will be using Linear Regression to predict `Gross` based on available *numeric* variables.

```
# Numeric ======================== Build & evaluate model 1
# (numeric variables only) This function randomizes data and
# splits into train and test datasets
pre_process_data = function(df, test_frac = 0.2) {
    # shuffle data - sample(nrow(df)) gives randomized indices
    # for selection
    df = df[sample(nrow(df)), ]
```

```
    num_samples = nrow(df)
    num_test_samples = as.integer(num_samples * test_frac)
    num_train_samples = num_samples - num_test_samples
    train_data = df[1:num_train_samples, ]
    test_data = df[(num_train_samples + 1):num_samples, ]
    return(list(train_data = train_data, test_data = test_data))
}

# get data processed and ready.
rslt = pre_process_data(df)
train_data = rslt$train_data
test_data = rslt$test_data
```

For our prediction, we will focus on numeric variables. Use the filter function which gets all types of numeric variabels including floats and ints.

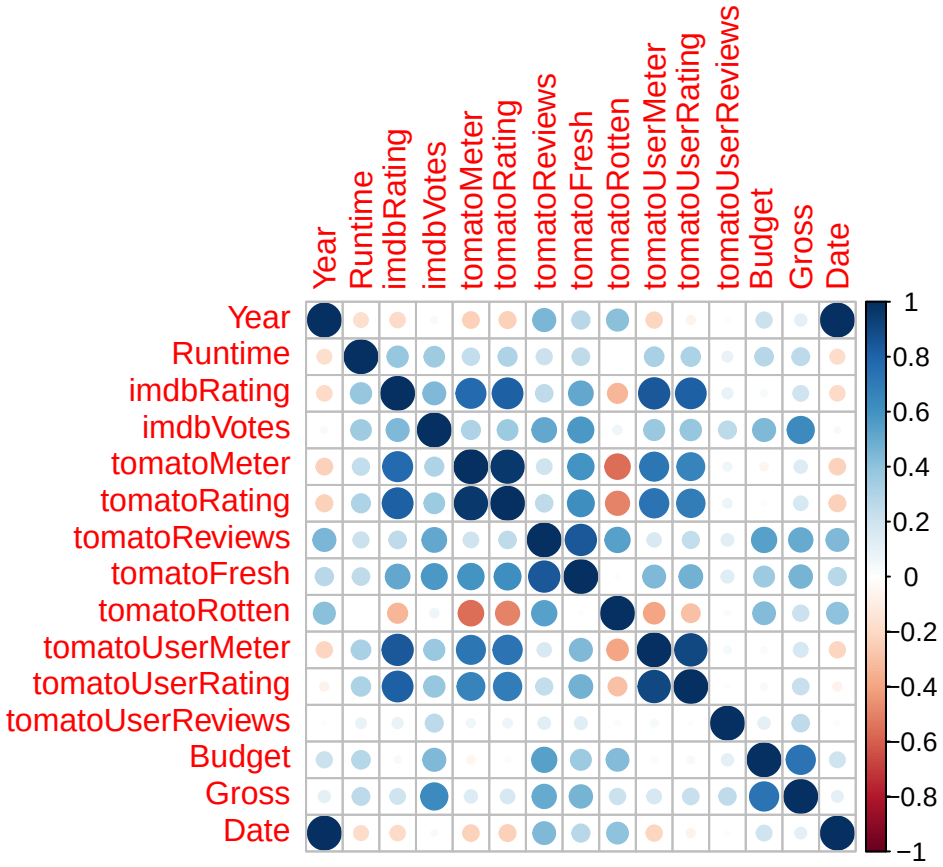
```
train_data_numeric = Filter(is.numeric, train_data)
test_data_numeric = Filter(is.numeric, test_data)

# replace some of the numeric variables that have NA in their
# observations with the column mean
train_data_numeric$imdbRating[is.na(train_data_numeric$imdbRating)] <- mean(train_data_numeric$imdbRati
    na.rm = TRUE)
train_data_numeric$Runtime[is.na(train_data_numeric$Runtime)] <- mean(train_data_numeric$Runtime,
    na.rm = TRUE)
train_data_numeric$tomatoMeter[is.na(train_data_numeric$tomatoMeter)] <- mean(train_data_numeric$tomatoM
    na.rm = TRUE)
train_data_numeric$tomatoRating[is.na(train_data_numeric$tomatoRating)] <- mean(train_data_numeric$toma
    na.rm = TRUE)
train_data_numeric$tomatoReviews[is.na(train_data_numeric$tomatoReviews)] <- mean(train_data_numeric$to
    na.rm = TRUE)
train_data_numeric$tomatoFresh[is.na(train_data_numeric$tomatoFresh)] <- mean(train_data_numeric$tomatoF
    na.rm = TRUE)
train_data_numeric$tomatoUserMeter[is.na(train_data_numeric$tomatoUserMeter)] <- mean(train_data_numeri
    na.rm = TRUE)
train_data_numeric$tomatoUserRating[is.na(train_data_numeric$tomatoUserRating)] <- mean(train_data_nume
    na.rm = TRUE)

# Let us also ensure that the dataset is free from NA values
train_data_numeric_nonNA <- na.omit(train_data_numeric)
test_data_numeric_nonNA <- na.omit(test_data_numeric)
```

We might also want to see if we should use only a subset of numeric variables to better understand the relationship between features. To do this, we could do a correlation plot. This plot will show correlation between -1 and +1. Very strongly positively correlated ones will be shown as very dark blue circle and the intensity of blue will smaller as the +ve correlation decreases. -ve correlation is shown in orange similarly.

```
tr_cor_df = cor(train_data_numeric_nonNA)
corrplot(tr_cor_df, method = "circle")
```



Based on the above, I chose only Runtime, imdbRating, imdbVotes, tomatoReviews, tomatoFresh, and Budget. I will drop the rest of the columns.

```
train_data_numeric_nonNA_subset = subset(train_data_numeric_nonNA,
    select = -c(tomatoRotten, Year, Date, tomatoMeter, tomatoUserMeter,
        tomatoUserReviews, tomatoRating))
test_data_numeric_nonNA_subset = subset(test_data_numeric_nonNA,
    select = -c(tomatoRotten, Year, Date, tomatoMeter, tomatoUserMeter,
        tomatoUserReviews, tomatoRating))
```

Function for performing linear regression using linear model(lm) command and gathering model stats

Let us create a kitchen sink model and evaluate it

```
kitchenSinkModel <- lm(Gross ~ . + .:., data = train_data_numeric_nonNA)
print(summary(kitchenSinkModel))
```

```
##
## Call:
## lm(formula = Gross ~ . + .:., data = train_data_numeric_nonNA)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -505979775  -26744696   -3559946   16799592  935055559
##
## Coefficients: (12 not defined because of singularities)
```

```
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    4.269e+10  3.954e+10   1.080 0.280361
## Year                           1.987e+07  5.804e+07   0.342 0.732070
## Runtime                        5.279e+07  1.293e+07   4.084 4.54e-05 ***
## imdbRating                     8.448e+08  8.502e+08   0.994 0.320473
## imdbVotes                      4.608e+03  4.826e+03   0.955 0.339750
## tomatoMeter                   -2.886e+07  3.718e+07  -0.776 0.437665
## tomatoRating                  -7.485e+07  7.348e+08  -0.102 0.918861
## tomatoReviews                 -2.463e+07  2.427e+07  -1.015 0.310163
## tomatoFresh                    7.322e+07  3.467e+07   2.112 0.034747 *
## tomatoRotten                          NA         NA      NA       NA
## tomatoUserMeter                4.172e+07  5.067e+07   0.823 0.410298
## tomatoUserRating              -4.418e+09  2.018e+09  -2.189 0.028672 *
## tomatoUserReviews              1.271e+03  3.684e+02   3.449 0.000570 ***
## Budget                         2.695e+01  2.377e+01   1.134 0.257002
## Date                          -6.111e+07  5.794e+07  -1.055 0.291616
## Year:Runtime                  -2.364e+05  1.377e+05  -1.717 0.086051 .
## Year:imdbRating               -3.999e+06  7.265e+06  -0.550 0.582038
## Year:imdbVotes                 1.013e+02  7.495e+01   1.352 0.176376
## Year:tomatoMeter               4.499e+05  3.874e+05   1.161 0.245642
## Year:tomatoRating             -2.654e+06  8.410e+06  -0.316 0.752370
## Year:tomatoReviews             1.709e+05  2.193e+05   0.779 0.435845
## Year:tomatoFresh              -2.490e+05  3.108e+05  -0.801 0.423004
## Year:tomatoRotten                     NA         NA      NA       NA
## Year:tomatoUserMeter           7.052e+04  5.422e+05   0.130 0.896539
## Year:tomatoUserRating         -2.476e+06  2.117e+07  -0.117 0.906893
## Year:tomatoUserReviews        -6.431e+01  5.408e+01  -1.189 0.234436
## Year:Budget                    3.588e-01  2.421e-01   1.482 0.138429
## Year:Date                      9.889e+03  9.571e+03   1.033 0.301612
## Runtime:imdbRating            -4.185e+05  2.234e+05  -1.873 0.061105 .
## Runtime:imdbVotes              2.892e+00  7.118e-01   4.063 4.96e-05 ***
## Runtime:tomatoMeter            1.306e+04  1.239e+04   1.054 0.291832
## Runtime:tomatoRating          -1.884e+05  2.386e+05  -0.790 0.429744
## Runtime:tomatoReviews          2.082e+03  3.859e+03   0.540 0.589476
## Runtime:tomatoFresh           -1.086e+03  4.952e+03  -0.219 0.826421
## Runtime:tomatoRotten                  NA         NA      NA       NA
## Runtime:tomatoUserMeter        2.246e+04  1.254e+04   1.791 0.073409 .
## Runtime:tomatoUserRating      -5.764e+05  4.639e+05  -1.242 0.214179
## Runtime:tomatoUserReviews      4.236e-02  2.922e-02   1.449 0.147302
## Runtime:Budget                -5.689e-03  2.523e-03  -2.255 0.024183 *
## Runtime:Date                   2.115e+05  1.361e+05   1.554 0.120218
## imdbRating:imdbVotes          -2.896e+02  5.072e+01  -5.710 1.24e-08 ***
## imdbRating:tomatoMeter         4.338e+05  3.907e+05   1.110 0.266942
## imdbRating:tomatoRating       -1.005e+07  6.684e+06  -1.503 0.132902
## imdbRating:tomatoReviews      -2.908e+05  1.271e+05  -2.288 0.022175 *
## imdbRating:tomatoFresh         2.654e+05  2.228e+05   1.191 0.233697
## imdbRating:tomatoRotten               NA         NA      NA       NA
## imdbRating:tomatoUserMeter    -9.168e+05  3.395e+05  -2.700 0.006968 **
## imdbRating:tomatoUserRating    2.205e+07  1.302e+07   1.693 0.090531 .
## imdbRating:tomatoUserReviews  -5.982e+00  2.105e+00  -2.842 0.004508 **
## imdbRating:Budget             -2.577e-01  1.450e-01  -1.777 0.075633 .
## imdbRating:Date                3.607e+06  7.225e+06   0.499 0.617640
## imdbVotes:tomatoMeter         -7.915e-01  3.424e+00  -0.231 0.817193
## imdbVotes:tomatoRating         8.838e+01  4.726e+01   1.870 0.061564 .
```

```
## imdbVotes:tomatoReviews            -2.702e+00  1.045e+00  -2.586 0.009764 **
## imdbVotes:tomatoFresh               1.167e+00  1.202e+00   0.971 0.331709
## imdbVotes:tomatoRotten                     NA         NA      NA       NA
## imdbVotes:tomatoUserMeter           5.311e+00  3.500e+00   1.517 0.129242
## imdbVotes:tomatoUserRating         -5.194e+02  9.380e+01  -5.537 3.32e-08 ***
## imdbVotes:tomatoUserReviews        -7.837e-06  3.719e-06  -2.107 0.035181 *
## imdbVotes:Budget                    4.072e-06  3.755e-07  10.844  < 2e-16 ***
## imdbVotes:Date                     -1.019e+02  7.479e+01  -1.363 0.172940
## tomatoMeter:tomatoRating            2.171e+05  1.486e+05   1.461 0.144004
## tomatoMeter:tomatoReviews          -1.051e+05  4.154e+04  -2.529 0.011485 *
## tomatoMeter:tomatoFresh            -2.836e+03  1.351e+04  -0.210 0.833723
## tomatoMeter:tomatoRotten                   NA         NA      NA       NA
## tomatoMeter:tomatoUserMeter        -3.699e+04  2.978e+04  -1.242 0.214343
## tomatoMeter:tomatoUserRating        4.053e+05  1.140e+06   0.355 0.722269
## tomatoMeter:tomatoUserReviews      -7.841e-01  1.552e-01  -5.052 4.63e-07 ***
## tomatoMeter:Budget                  9.139e-03  9.475e-03   0.965 0.334851
## tomatoMeter:Date                   -4.374e+05  3.837e+05  -1.140 0.254429
## tomatoRating:tomatoReviews          1.225e+05  1.760e+05   0.696 0.486597
## tomatoRating:tomatoFresh           -3.033e+05  2.300e+05  -1.319 0.187395
## tomatoRating:tomatoRotten                  NA         NA      NA       NA
## tomatoRating:tomatoUserMeter        1.049e+06  5.889e+05   1.782 0.074886 .
## tomatoRating:tomatoUserRating      -3.250e+07  2.201e+07  -1.477 0.139742
## tomatoRating:tomatoUserReviews      7.216e+00  2.094e+00   3.446 0.000575 ***
## tomatoRating:Budget                 8.322e-02  1.834e-01   0.454 0.649954
## tomatoRating:Date                   2.747e+06  8.342e+06   0.329 0.741941
## tomatoReviews:tomatoFresh           3.205e+03  1.285e+03   2.494 0.012673 *
## tomatoReviews:tomatoRotten          4.492e+03  1.895e+03   2.370 0.017844 *
## tomatoReviews:tomatoUserMeter      -2.991e+03  9.916e+03  -0.302 0.762924
## tomatoReviews:tomatoUserRating      8.656e+05  4.008e+05   2.160 0.030855 *
## tomatoReviews:tomatoUserReviews    -2.060e-01  4.621e-02  -4.459 8.53e-06 ***
## tomatoReviews:Budget               -1.948e-03  1.972e-03  -0.988 0.323233
## tomatoReviews:Date                 -1.596e+05  2.186e+05  -0.730 0.465438
## tomatoFresh:tomatoRotten           -5.191e+03  4.881e+03  -1.064 0.287599
## tomatoFresh:tomatoUserMeter        -2.009e+03  1.537e+04  -0.131 0.895981
## tomatoFresh:tomatoUserRating        1.426e+05  5.554e+05   0.257 0.797431
## tomatoFresh:tomatoUserReviews       3.401e-01  4.887e-02   6.958 4.16e-12 ***
## tomatoFresh:Budget                 -2.066e-03  2.698e-03  -0.766 0.443893
## tomatoFresh:Date                    2.176e+05  3.092e+05   0.704 0.481701
## tomatoRotten:tomatoUserMeter               NA         NA      NA       NA
## tomatoRotten:tomatoUserRating              NA         NA      NA       NA
## tomatoRotten:tomatoUserReviews             NA         NA      NA       NA
## tomatoRotten:Budget                        NA         NA      NA       NA
## tomatoRotten:Date                          NA         NA      NA       NA
## tomatoUserMeter:tomatoUserRating    1.029e+06  4.001e+05   2.573 0.010131 *
## tomatoUserMeter:tomatoUserReviews   5.577e-02  1.130e-01   0.493 0.621771
## tomatoUserMeter:Budget             -3.632e-02  9.274e-03  -3.916 9.18e-05 ***
## tomatoUserMeter:Date               -9.265e+04  5.364e+05  -0.173 0.862875
## tomatoUserRating:tomatoUserReviews  6.656e+00  3.265e+00   2.039 0.041549 *
## tomatoUserRating:Budget             3.015e+00  3.367e-01   8.954  < 2e-16 ***
## tomatoUserRating:Date               4.648e+06  2.094e+07   0.222 0.824352
## tomatoUserReviews:Budget            4.121e-08  1.557e-08   2.646 0.008179 **
## tomatoUserReviews:Date              6.368e+01  5.407e+01   1.178 0.238988
## Budget:Date                        -3.747e-01  2.411e-01  -1.555 0.120150
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79690000 on 3183 degrees of freedom
## Multiple R-squared:  0.7974, Adjusted R-squared:  0.7915
## F-statistic: 134.7 on 93 and 3183 DF,  p-value: < 2.2e-16
```

```
kitchenSinkModel.rmse <- sqrt(mean(kitchenSinkModel$residuals^2))
cat("Train RMSE is:", kitchenSinkModel.rmse, "\n")
```

```
## Train RMSE is: 78535793
```

```
# Cross validation
kitchenSinkModel_cv <- cvFit(kitchenSinkModel, data = train_data_numeric_nonNA,
    y = train_data_numeric_nonNA$Gross, K = 10)
```

```
## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading

## Warning in predict.lm(...): prediction from a rank-deficient fit may be
## misleading
```

```
print(kitchenSinkModel_cv)
```

```
## 10-fold CV results:
##       CV
## 88921659
```

Let us create a model and evaluate it

```r
numericModel <- lm(Gross ~ ., data = train_data_numeric_nonNA_subset)
print(summary(numericModel))
```

```
##
## Call:
## lm(formula = Gross ~ ., data = train_data_numeric_nonNA_subset)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -621590811  -33492878   -1259685   22413655 1424281921
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.885e+07  1.719e+07  -1.097    0.273
## Runtime          -4.090e+05  9.771e+04  -4.185 2.92e-05 ***
## imdbRating       -1.962e+07  3.184e+06  -6.163 8.03e-10 ***
## imdbVotes         4.362e+02  1.634e+01  26.689  < 2e-16 ***
## tomatoReviews    -3.536e+05  5.816e+04  -6.080 1.34e-09 ***
## tomatoFresh       4.860e+05  7.300e+04   6.657 3.26e-11 ***
## tomatoUserRating  5.092e+07  6.664e+06   7.641 2.81e-14 ***
## Budget            2.531e+00  5.539e-02  45.686  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100100000 on 3269 degrees of freedom
## Multiple R-squared:  0.6719, Adjusted R-squared:  0.6712
## F-statistic: 956.4 on 7 and 3269 DF,  p-value: < 2.2e-16
```

```r
numericModel.rmse <- sqrt(mean(numericModel$residuals^2))
cat("Train RMSE is:", numericModel.rmse, "\n")
```

```
## Train RMSE is: 99951574
```

```r
# Cross validation
numericModel_cv <- cvFit(numericModel, data = train_data_numeric_nonNA_subset,
    y = train_data_numeric_nonNA_subset$Gross, K = 10)
print(numericModel_cv)
```

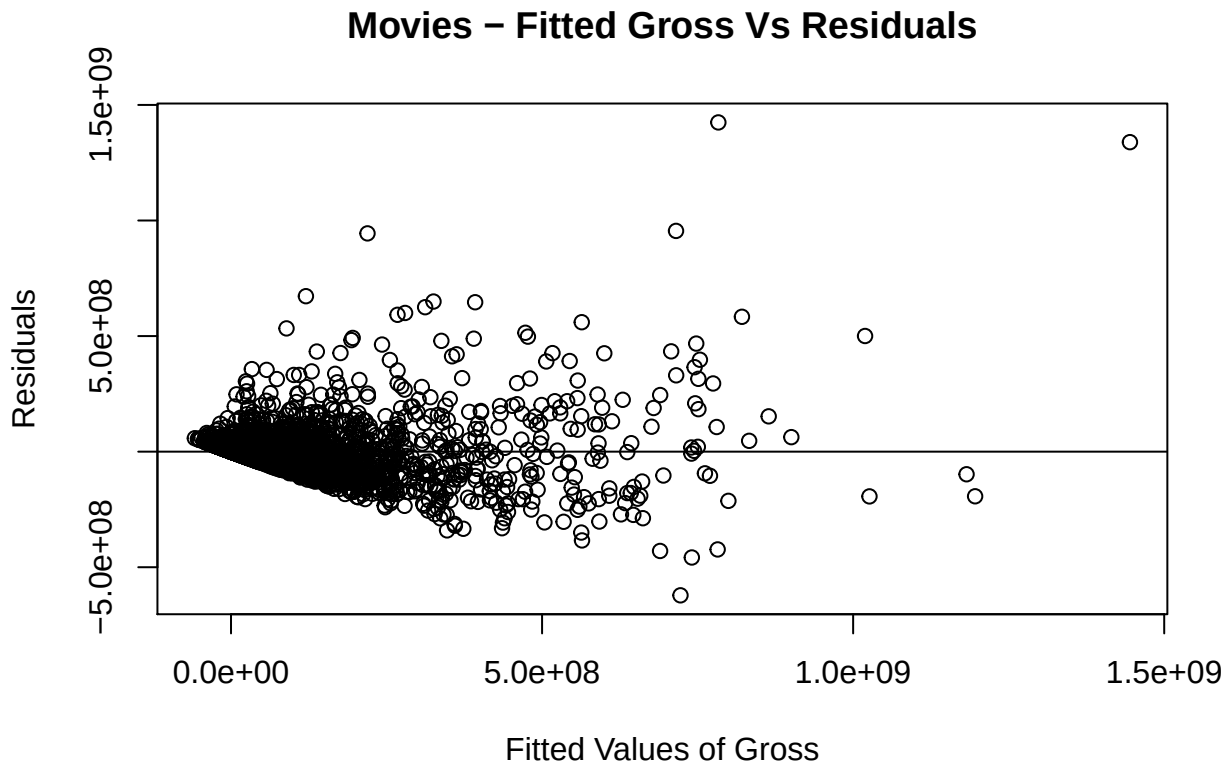
```
## 10-fold CV results:
##        CV
## 100616392
```

Transformations

Let us see if we can improve the prediction quality from basic Numerical variables as much as possible by adding feature transformations of the numeric variables. We will explore both numeric transformations such as power transforms and non-numeric transformations of the numeric variables like binning (e.g. `is_budget_greater_than_3M`).

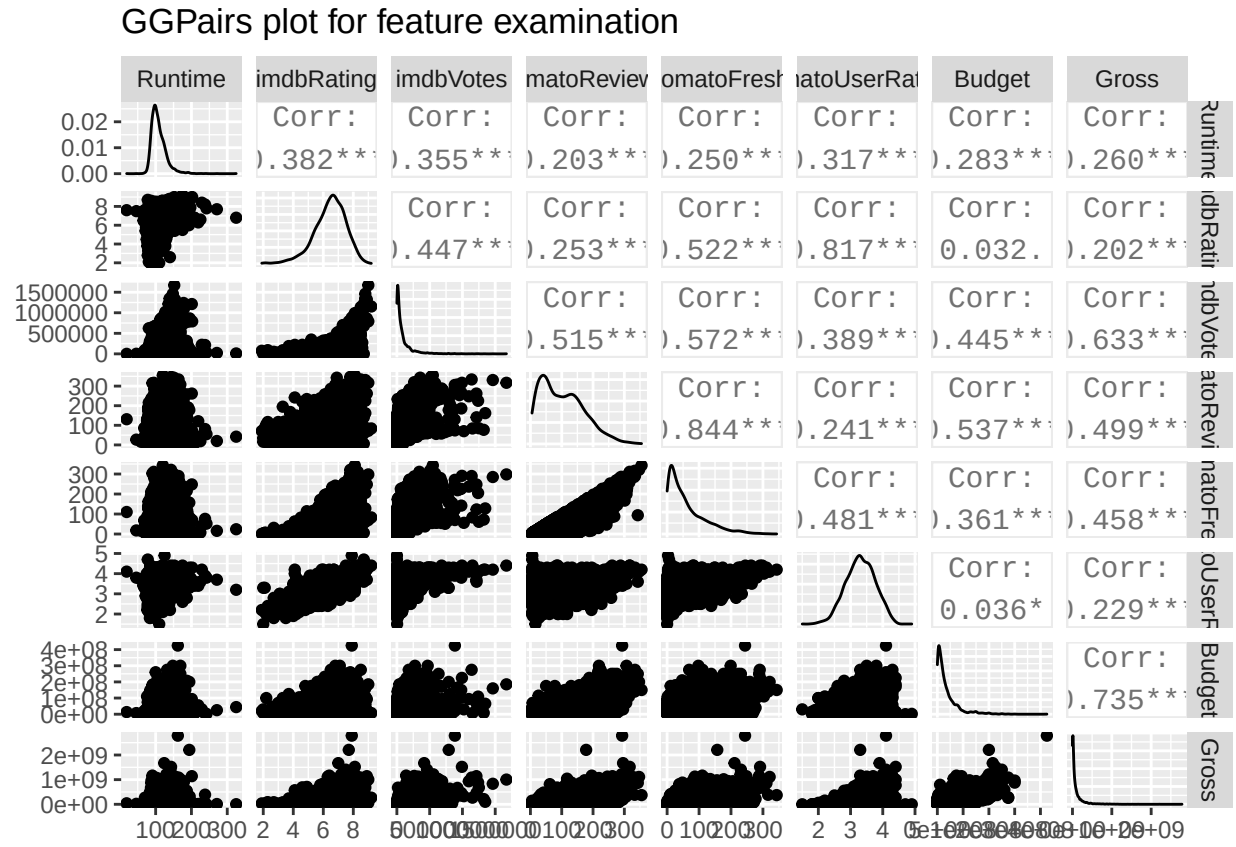
Let us plot the residuals to see if there is non-linearity in features at a gross level. We can do this by plotting residuals for Gross

```
mdl = lm(Gross ~ ., data = train_data_numeric_nonNA_subset)
m.res = resid(mdl)
plot(mdl$fitted.values, m.res, ylab = "Residuals", xlab = "Fitted Values of Gross",
    main = "Movies - Fitted Gross Vs Residuals")
abline(0, 0)
```



There definitely exists a correlation as is evident in the residual plot above. It does not look randmly scattered as we would expect in a fully linearly fit model. It appears **biased** and **heteroscedastic**. To examine where the relations might be, let us do pairwise plot

```
df_temp = subset(train_data_numeric_nonNA_subset, train_data_numeric_nonNA_subset$Gross !=
    0)
suppressWarnings(ggpairs(df_temp)) + ggtitle("GGPairs plot for feature examination")
```




```
# Model 2. Transformed Numeric Variables
# ====================================== Build & evaluate
# model 2 (transformed numeric variables only)

# feature transformation functions split a feature at mean
# value to bin into high or low (1 or 0)
feature_xform_to_binary = function(df, feature) {
    feature_mean_val = mean(df[, feature])
    new_label = paste0(feature, "_bin")
    # assign all to 0
    df[new_label] = 0
    df[df[, feature] > feature_mean_val, new_label] = 1
    return(df)
}
```

Based on examining the pairwise plots we choose the variables to transform.

```
feature_transform = function(df) {
    df = na.omit(df)
    df = feature_xform_to_binary(df, "imdbVotes")  #imdbVotes_bin - binary xform
    df = feature_xform_to_binary(df, "Budget")  #Budget_bin - binary xform

    df["Runtime_log"] = log(df$Runtime)  # log xform

    df["imdbRating_P2"] = (df$imdbRating)^2  # power transform
    df["imdbVotes_P2"] = (df$imdbVotes)^2  # power transform
```

```
    df["tomatoUserRating_P2"] = (df$tomatoUserRating)^2  # power transform
    return(df)
}

train_data_xform = feature_transform(train_data_numeric_nonNA_subset)
test_data_xform = feature_transform(test_data_numeric_nonNA_subset)

# call function to compute average training and test rsmes
# for range of sizes
resp_var = "Gross"
fx = paste(resp_var, " ~ ", ".", " - ", resp_var)

# Fit a model and print results
xformModel <- lm(fx, data = train_data_xform)
print(summary(xformModel))
```

```
##
## Call:
## lm(formula = fx, data = train_data_xform)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -499045626  -33764213   -3793668   25404404 1361418915
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3.789e+08  2.013e+08   1.883 0.059853 .
## Runtime              1.382e+06  4.418e+05   3.127 0.001780 **
## imdbRating           7.566e+07  1.677e+07   4.510 6.71e-06 ***
## imdbVotes            7.107e+02  4.419e+01  16.084  < 2e-16 ***
## tomatoReviews       -4.597e+05  6.060e+04  -7.586 4.27e-14 ***
## tomatoFresh          5.333e+05  7.447e+04   7.160 9.89e-13 ***
## tomatoUserRating     8.877e+07  5.197e+07   1.708 0.087717 .
## Budget               2.696e+00  6.908e-02  39.029  < 2e-16 ***
## imdbVotes_bin       -2.630e+06  6.697e+06  -0.393 0.694587
## Budget_bin          -3.989e+07  5.463e+06  -7.302 3.55e-13 ***
## Runtime_log         -1.939e+08  5.158e+07  -3.759 0.000174 ***
## imdbRating_P2       -8.815e+06  1.440e+06  -6.121 1.04e-09 ***
## imdbVotes_P2        -2.282e-04  3.672e-05  -6.215 5.77e-10 ***
## tomatoUserRating_P2 -4.627e+06  7.932e+06  -0.583 0.559677
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97810000 on 3263 degrees of freedom
## Multiple R-squared:  0.6872, Adjusted R-squared:  0.6859
## F-statistic: 551.4 on 13 and 3263 DF,  p-value: < 2.2e-16
```

```
xformModel.rmse <- sqrt(mean(xformModel$residuals^2))
cat("Train RMSE is:", xformModel.rmse, "\n")
```

```
## Train RMSE is: 97596497
```

```r
# Cross validation results
xformModel_cv <- cvFit(xformModel, data = train_data_xform, y = train_data_xform$Gross,
    K = 10)
print(xformModel_cv)
```

```
## 10-fold CV results:
##       CV
## 99667560
```

Interactions

```r
intModel <- lm(Gross ~ . + .:., data = train_data_numeric_nonNA_subset)
print(summary(intModel))
```

```
##
## Call:
## lm(formula = Gross ~ . + .:., data = train_data_numeric_nonNA_subset)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -565819859  -27968515   -3584266   16367670  942941099
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -2.262e+08  9.744e+07  -2.322 0.020309 *
## Runtime                         2.452e+06  8.940e+05   2.742 0.006136 **
## imdbRating                      2.509e+07  1.955e+07   1.283 0.199464
## imdbVotes                       4.023e+03  1.896e+02  21.216  < 2e-16 ***
## tomatoReviews                  -6.859e+05  4.887e+05  -1.404 0.160501
## tomatoFresh                     2.450e+05  6.279e+05   0.390 0.696390
## tomatoUserRating                1.964e+07  4.051e+07   0.485 0.627884
## Budget                         -2.789e+00  5.490e-01  -5.080 3.99e-07 ***
## Runtime:imdbRating             -6.563e+04  1.688e+05  -0.389 0.697411
## Runtime:imdbVotes               4.090e+00  6.362e-01   6.429 1.47e-10 ***
## Runtime:tomatoReviews          -6.858e+03  3.157e+03  -2.172 0.029898 *
## Runtime:tomatoFresh             5.229e+03  3.821e+03   1.369 0.171224
## Runtime:tomatoUserRating       -5.846e+05  3.487e+05  -1.677 0.093682 .
## Runtime:Budget                 -5.978e-03  2.445e-03  -2.445 0.014532 *
## imdbRating:imdbVotes           -3.030e+02  3.180e+01  -9.527  < 2e-16 ***
## imdbRating:tomatoReviews       -5.472e+05  8.549e+04  -6.400 1.77e-10 ***
## imdbRating:tomatoFresh          3.865e+05  1.102e+05   3.508 0.000458 ***
## imdbRating:tomatoUserRating    -6.974e+05  3.725e+06  -0.187 0.851502
## imdbRating:Budget               3.756e-02  1.072e-01   0.351 0.725935
## imdbVotes:tomatoReviews        -5.152e+00  6.104e-01  -8.441  < 2e-16 ***
## imdbVotes:tomatoFresh           4.494e+00  6.666e-01   6.742 1.84e-11 ***
## imdbVotes:tomatoUserRating     -4.152e+02  6.137e+01  -6.766 1.56e-11 ***
## imdbVotes:Budget                3.567e-06  3.255e-07  10.961  < 2e-16 ***
## tomatoReviews:tomatoFresh      -1.440e+02  6.536e+02  -0.220 0.825648
## tomatoReviews:tomatoUserRating  1.483e+06  1.984e+05   7.476 9.82e-14 ***
## tomatoReviews:Budget            3.631e-03  1.225e-03   2.964 0.003059 **
## tomatoFresh:tomatoUserRating   -9.931e+05  2.367e+05  -4.196 2.79e-05 ***
```

```
## tomatoFresh:Budget             -1.370e-03  1.512e-03  -0.906 0.365143
## tomatoUserRating:Budget         1.198e+00  2.037e-01   5.881 4.50e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 84690000 on 3248 degrees of freedom
## Multiple R-squared:  0.7665, Adjusted R-squared:  0.7645
## F-statistic: 380.8 on 28 and 3248 DF,  p-value: < 2.2e-16
```

```
intModel.rmse <- sqrt(mean(intModel$residuals^2))
cat("Train RMSE for Interaction model is:", intModel.rmse, "\n")
```

```
## Train RMSE for Interaction model is: 84316108
```

```
# Cross validation results
intModel_cv <- cvFit(intModel, data = train_data_numeric_nonNA_subset,
    y = train_data_numeric_nonNA_subset$Gross, K = 10)
print(intModel_cv)
```

```
## 10-fold CV results:
##       CV
## 88411641
```

Final model

transforms + interactions added to the existing numeric variables

```
finalModel <- lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
    Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
    Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh + imdbVotes:tomatoReviews +
    imdbVotes:tomatoFresh + imdbVotes:Budget + tomatoReviews:imdbRating_P2 +
    tomatoReviews:tomatoUserRating_P2 + tomatoUserRating:Runtime_log +
    Budget:imdbVotes_bin + Budget:Runtime_log + Runtime_log:imdbVotes_P2 +
    Runtime_log:tomatoUserRating_P2, data = train_data_xform)
print(summary(finalModel))
```

```
##
## Call:
## lm(formula = Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
##     Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
##     Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh + imdbVotes:tomatoReviews +
##     imdbVotes:tomatoFresh + imdbVotes:Budget + tomatoReviews:imdbRating_P2 +
##     tomatoReviews:tomatoUserRating_P2 + tomatoUserRating:Runtime_log +
##     Budget:imdbVotes_bin + Budget:Runtime_log + Runtime_log:imdbVotes_P2 +
##     Runtime_log:tomatoUserRating_P2, data = train_data_xform)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -714775683  -28658655   -4505395   16224652  919358345
##
## Coefficients:
```

```
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -2.230e+10  1.053e+10  -2.118 0.034282 *
## Runtime                           -7.023e+07  2.659e+07  -2.641 0.008311 **
## imdbRating                         2.890e+07  1.714e+07   1.686 0.091936 .
## imdbVotes                          7.423e+02  5.172e+01  14.352  < 2e-16 ***
## tomatoReviews                     -1.907e+05  1.805e+05  -1.056 0.291037
## tomatoFresh                        6.311e+05  5.840e+05   1.081 0.279938
## tomatoUserRating                   1.282e+10  5.948e+09   2.155 0.031209 *
## Budget                             3.627e+01  7.051e+00   5.143 2.86e-07 ***
## imdbVotes_bin                     -5.569e+06  7.663e+06  -0.727 0.467456
## Budget_bin                         1.159e+08  2.763e+07   4.193 2.82e-05 ***
## Runtime_log                        6.332e+09  2.863e+09   2.211 0.027071 *
## imdbRating_P2                     -1.570e+06  1.548e+06  -1.014 0.310565
## imdbVotes_P2                       1.839e-02  2.892e-03   6.361 2.29e-10 ***
## tomatoUserRating_P2               -1.845e+09  8.347e+08  -2.211 0.027133 *
## Runtime:tomatoFresh                4.759e+03  1.795e+03   2.652 0.008051 **
## Runtime:tomatoUserRating           4.161e+07  1.487e+07   2.798 0.005179 **
## Runtime:Budget                     7.312e-02  1.506e-02   4.854 1.27e-06 ***
## Runtime:Budget_bin                -9.934e+05  2.401e+05  -4.138 3.59e-05 ***
## Runtime:imdbVotes_P2               3.588e-05  4.763e-06   7.532 6.44e-14 ***
## Runtime:tomatoUserRating_P2       -6.119e+06  2.071e+06  -2.954 0.003155 **
## imdbRating:tomatoFresh            -1.088e+05  9.161e+04  -1.187 0.235155
## imdbVotes:tomatoReviews           -1.654e+00  4.641e-01  -3.564 0.000371 ***
## imdbVotes:tomatoFresh              7.253e-01  5.286e-01   1.372 0.170128
## imdbVotes:Budget                   5.414e-06  2.978e-07  18.177  < 2e-16 ***
## tomatoReviews:imdbRating_P2       -3.628e+04  5.124e+03  -7.081 1.75e-12 ***
## tomatoReviews:tomatoUserRating_P2  1.370e+05  1.329e+04  10.305  < 2e-16 ***
## tomatoUserRating:Runtime_log      -3.676e+09  1.613e+09  -2.278 0.022763 *
## Budget:imdbVotes_bin               2.897e-01  1.196e-01   2.422 0.015494 *
## Budget:Runtime_log                -9.280e+00  1.851e+00  -5.014 5.62e-07 ***
## Runtime_log:imdbVotes_P2          -4.842e-03  7.157e-04  -6.765 1.57e-11 ***
## Runtime_log:tomatoUserRating_P2    5.309e+08  2.260e+08   2.349 0.018875 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85940000 on 3246 degrees of freedom
## Multiple R-squared:  0.7597, Adjusted R-squared:  0.7575
## F-statistic: 342.1 on 30 and 3246 DF,  p-value: < 2.2e-16
```

```
finalModel.rmse <- sqrt(mean(finalModel$residuals^2))
cat("Train RMSE for Final model is:", finalModel.rmse, "\n")
```

```
## Train RMSE for Final model is: 85532912
```

```
# Cross validation results
finalModel_cv <- cvFit(finalModel, data = train_data_xform, y = train_data_numeric_nonNA_subset$Gross,
    K = 10)
print(finalModel_cv)
```

```
## 10-fold CV results:
##       CV
## 89282807
```

Lasso model

```
# #train_temp <- subset(train_data_xform, select=-c(Gross))
# baseline_log_linear<-lm(Gross~., data = train_data_xform)
# print(summary(baseline_log_linear)) # set lambda sequence
# to use for lasso #lambdas = 10^seq(-10,1.5,0.1) lambdas =
# 10^seq(-2,1.5,0.1) #lasso # train_Lasso <-
# train_data_numeric_nonNA_subset[,log_transform] fm.lasso =
# glmnet(as.matrix(train_data_xform),
# as.double(train_data_xform$Gross), alpha = 1, lambda =
# lambdas, standardize = TRUE, thresh = 1e-12)
# summary(fm.lasso)
# cv_fit<-cv.glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, lambda = lambdas, standardize =
# TRUE) plot(cv_fit) opt_lambda<-cv_fit$lambda.min
# print(opt_lambda) #summary(cv_fit) #plot(fm.lasso, xvar
# ='lambda') lasso.predict =
# predict(glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, alpha = 1, lambda = opt_lambda), s
# = opt_lambda, newx = as.matrix(train_data_xform))
# sqrt(mean((lasso.predict - train_data_xform$Gross)^2))
```

```
# # Different approach to calculate Lasso RMSE lasso_reg
# <-cv.glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, alpha = 1, lambda = lambdas,
# standardize = TRUE, nfolds = 5) lambda_best <-
# lasso_reg$lambda.min lambda_best
```

```
# lasso_model <- glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, alpha = 1, lambda = lambda_best,
# standardize = TRUE) prediction_train <-
# predict(lasso_model, s = lambda_best, newx =
# as.matrix(train_data_xform)) sqrt(mean((prediction_train -
# train_data_xform$Gross)^2)) ##RMSE
```

```
# fm.ridge = glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, alpha = 0, lambda = lambdas,
# standardize = TRUE) #summary(fm.ridge) #plot(fm.ridge, xvar
# ='lambda')
# cv_fit_ridge<-cv.glmnet(as.matrix(train_data_xform),
# train_data_xform$Gross, lambda = lambdas, nfolds = 10)
# plot(cv_fit_ridge) opt_lambda<-cv_fit_ridge$lambda.min
# print(opt_lambda)
```

```
# index_RF = sample(nrow(train_data_xform), size =
# nrow(train_data_xform)*0.60) train_data[index_RF,]
# fit=randomForest(Gross~., data=train_data_xform, importance
# = TRUE) #png(file='C:/Users/uzair/Desktop/Stanford/MS&E
# 226/Project/RF.png', type = 'cairo') varImpPlot(fit)
# print(fit)
```

```
# rmse_RF = sqrt(mean((predict(fit, train_data_xform) -
# train_data_xform$Gross)^2)) print(rmse_RF)
```

```
# index_RF_m = sample(nrow(train_data_numeric_nonNA), size =
# nrow(train_data_numeric_nonNA)) train_data[index_RF_m,]
# fit1=randomForest(Gross~., data=train_data_numeric_nonNA)
# varImpPlot(fit1) print(fit1)
```

Part 2

```
finModel.test <- lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
    Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
    Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh + imdbVotes:tomatoReviews +
    imdbVotes:tomatoFresh + imdbVotes:Budget + tomatoReviews:imdbRating_P2 +
    tomatoReviews:tomatoUserRating_P2 + tomatoUserRating:Runtime_log +
    Budget:imdbVotes_bin + Budget:Runtime_log + Runtime_log:imdbVotes_P2 +
    Runtime_log:tomatoUserRating_P2, data = test_data_xform)
print(summary(finModel.test))
```

```
##
## Call:
## lm(formula = Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
##     Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
##     Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh + imdbVotes:tomatoReviews +
##     imdbVotes:tomatoFresh + imdbVotes:Budget + tomatoReviews:imdbRating_P2 +
##     tomatoReviews:tomatoUserRating_P2 + tomatoUserRating:Runtime_log +
##     Budget:imdbVotes_bin + Budget:Runtime_log + Runtime_log:imdbVotes_P2 +
##     Runtime_log:tomatoUserRating_P2, data = test_data_xform)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -296897118  -31230174   -3536793   20273093  825205946
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     7.147e+09  2.806e+10   0.255 0.799052
## Runtime                         1.557e+06  7.182e+07   0.022 0.982714
## imdbRating                      9.560e+07  2.928e+07   3.265 0.001143 **
## imdbVotes                       8.527e+02  1.183e+02   7.208 1.34e-12 ***
## tomatoReviews                   3.484e+04  3.275e+05   0.106 0.915321
## tomatoFresh                     7.926e+05  9.458e+05   0.838 0.402258
## tomatoUserRating               -5.013e+09  1.632e+10  -0.307 0.758769
## Budget                          2.445e+01  1.494e+01   1.636 0.102156
## imdbVotes_bin                  -1.667e+07  1.703e+07  -0.979 0.327981
## Budget_bin                      1.205e+08  5.583e+07   2.159 0.031148 *
## Runtime_log                    -1.649e+09  7.651e+09  -0.216 0.829388
## imdbRating_P2                  -7.929e+06  2.707e+06  -2.929 0.003502 **
## imdbVotes_P2                   -1.228e-02  1.348e-02  -0.911 0.362830
## tomatoUserRating_P2             7.323e+08  2.351e+09   0.312 0.755483
## Runtime:tomatoFresh            -6.365e+03  3.478e+03  -1.830 0.067587 .
## Runtime:tomatoUserRating       -2.434e+06  4.121e+07  -0.059 0.952912
## Runtime:Budget                  6.903e-02  3.315e-02   2.082 0.037637 *
```

```
## Runtime:Budget_bin                -1.234e+06  4.877e+05  -2.529 0.011622 *
## Runtime:imdbVotes_P2              -2.144e-05  2.970e-05  -0.722 0.470512
## Runtime:tomatoUserRating_P2        4.327e+05  5.854e+06   0.074 0.941090
## imdbRating:tomatoFresh             6.408e+04  1.459e+05   0.439 0.660594
## imdbVotes:tomatoReviews           -1.015e+00  9.906e-01  -1.025 0.305777
## imdbVotes:tomatoFresh              7.025e-01  1.097e+00   0.640 0.522040
## imdbVotes:Budget                   1.348e-06  7.008e-07   1.924 0.054719 .
## tomatoReviews:imdbRating_P2       -3.965e+04  9.148e+03  -4.334 1.65e-05 ***
## tomatoReviews:tomatoUserRating_P2  1.101e+05  2.482e+04   4.437 1.04e-05 ***
## tomatoUserRating:Runtime_log       1.145e+09  4.435e+09   0.258 0.796418
## Budget:imdbVotes_bin               8.821e-01  2.581e-01   3.418 0.000664 ***
## Budget:Runtime_log                -6.542e+00  3.956e+00  -1.654 0.098581 .
## Runtime_log:imdbVotes_P2           2.994e-03  3.568e-03   0.839 0.401594
## Runtime_log:tomatoUserRating_P2   -1.692e+08  6.368e+08  -0.266 0.790520
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 80970000 on 785 degrees of freedom
## Multiple R-squared:  0.7453, Adjusted R-squared:  0.7355
## F-statistic: 76.56 on 30 and 785 DF,  p-value: < 2.2e-16
```

```
finModel.rmse.test <- sqrt(mean(finModel.test$residuals^2))
cat("Test RMSE for Interaction model is:", finModel.rmse.test,
    "\n")
```

```
## Test RMSE for Interaction model is: 79418521
```

```
# Cross validation results
finModel_cv_test <- cvFit(finModel.test, data = test_data_xform,
    y = test_data_xform$Gross, K = 10)
print(finModel_cv_test)
```

```
## 10-fold CV results:
##        CV
## 103985820
```

#####Define Bootstrap function#####

```
boot.function = function(train_data_xform, index) {
    return(coef(lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
        Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
        Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh +
        imdbVotes:tomatoReviews + imdbVotes:tomatoFresh + imdbVotes:Budget +
        tomatoReviews:imdbRating_P2 + tomatoReviews:tomatoUserRating_P2 +
        tomatoUserRating:Runtime_log + Budget:imdbVotes_bin +
        Budget:Runtime_log + Runtime_log:imdbVotes_P2 + Runtime_log:tomatoUserRating_P2,
        data = train_data_xform, subset = index)))
}

# Fitting linear models to 1000 bootstrapped samples of
# training data to estimate Standard error of coefficients
```

```r
Iteration <- 1000
set.seed(3945827)

lm.bootstrap <- boot(data = train_data_xform, statistic = boot.function,
    R = Iteration)
print(lm.bootstrap)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = train_data_xform, statistic = boot.function, R = Iteration)
##
##
## Bootstrap Statistics :
##           original        bias     std. error
## t1*  -2.230166e+10  3.721901e+09 1.483520e+10
## t2*  -7.022694e+07  1.175163e+07 4.098417e+07
## t3*   2.889604e+07 -2.464760e+05 1.415596e+07
## t4*   7.423267e+02 -7.921896e+00 1.010138e+02
## t5*  -1.906537e+05 -8.624389e+03 2.345939e+05
## t6*   6.311184e+05  4.442292e+04 7.667117e+05
## t7*   1.281953e+10 -1.870421e+09 8.422273e+09
## t8*   3.626551e+01  5.352337e-01 1.770653e+01
## t9*  -5.568787e+06  1.883500e+05 1.153466e+07
## t10*  1.158627e+08 -2.916405e+06 4.190033e+07
## t11*  6.332062e+09 -1.065340e+09 4.113302e+09
## t12* -1.570106e+06  5.168710e+04 1.415050e+06
## t13*  1.839342e-02 -6.475640e-04 7.175284e-03
## t14* -1.845163e+09  2.311002e+08 1.203111e+09
## t15*  4.758844e+03 -3.032923e+02 2.932615e+03
## t16*  4.161062e+07 -6.147607e+06 2.320327e+07
## t17*  7.311891e-02  7.971708e-04 4.069408e-02
## t18* -9.934075e+05  2.300654e+04 3.749996e+05
## t19*  3.587679e-05 -1.313095e-06 1.256399e-05
## t20* -6.118751e+06  7.999018e+05 3.304292e+06
## t21* -1.087762e+05 -3.476231e+03 1.203586e+05
## t22* -1.653959e+00  1.197048e-01 9.411329e-01
## t23*  7.253442e-01 -4.672268e-02 1.002068e+00
## t24*  5.413600e-06 -3.491279e-09 9.775637e-07
## t25* -3.628106e+04  2.916194e+02 7.069645e+03
## t26*  1.369971e+05 -4.389648e+02 2.220633e+04
## t27* -3.675732e+09  5.408659e+08 2.333464e+09
## t28*  2.897452e-01 -4.470593e-03 2.283372e-01
## t29* -9.280389e+00 -1.334807e-01 4.730266e+00
## t30* -4.842240e-03  1.664656e-04 1.802166e-03
## t31*  5.309012e+08 -6.774534e+07 3.330722e+08
```

```r
# reg_table <- data.frame(summary(finalModel)$coefficients)
# %>% bind_cols(data_frame(variable =
# names(finalModel$coefficients))) %>% transmute(variable =
# variable, confint_2.5_reg = Estimate - 1.96 * Std..Error,
```

```
# original_reg = Estimate, confint_97.5_reg = Estimate + 1.96
# * Std..Error)
```

```
# boostrap_table <- summary(lm.bootstrap) %>%
# bind_cols(data_frame(variable =
# names(finalModel$coefficients))) %>%
# mutate(confint_2.5_boot = original - bootSE,
# confint_97.5_boot = original + bootSE, original_boot =
# original) %>% select(variable, confint_2.5_boot,
# original_boot, confint_97.5_boot)
```

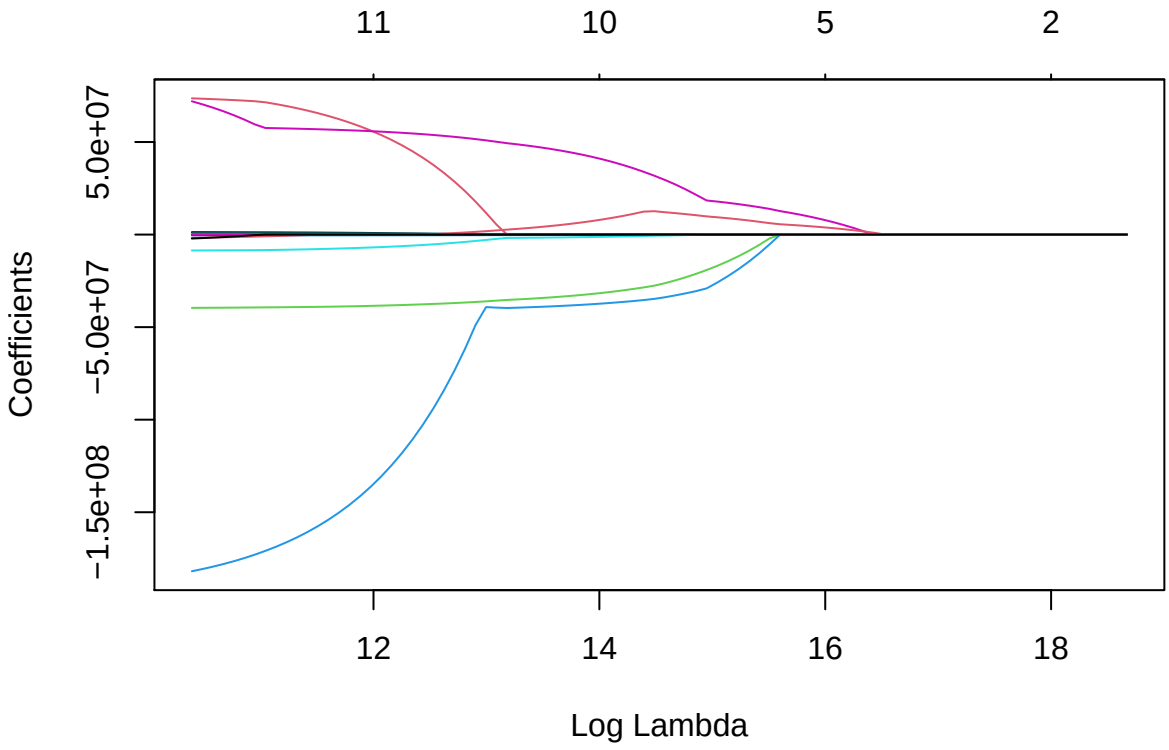
```
X <- train_data_xform
X$Gross <- NULL
Y <- train_data_xform$Gross
df = data.frame(X, Y)

coef.boot = function(data, indices) {
    fm = lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
        Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
        Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh +
        imdbVotes:tomatoReviews + imdbVotes:tomatoFresh + imdbVotes:Budget +
        tomatoReviews:imdbRating_P2 + tomatoReviews:tomatoUserRating_P2 +
        tomatoUserRating:Runtime_log + Budget:imdbVotes_bin +
        Budget:Runtime_log + Runtime_log:imdbVotes_P2 + Runtime_log:tomatoUserRating_P2,
        data = train_data_xform)
    return(coef(fm))
}


boot.out = boot(df, coef.boot, 1000)
summary(boot.out)
```

```
##           Length Class      Mode
## t0           31  -none-     numeric
## t         31000  -none-     numeric
## R             1  -none-     numeric
## data         14  data.frame list
## seed        626  -none-     numeric
## statistic     1  -none-     function
## sim           1  -none-     character
## call          4  -none-     call
## stype         1  -none-     character
## strata     3277  -none-     numeric
## weights    3277  -none-     numeric
```

```
lambdas = 10^seq(-10, 2, 0.1)
train_Lasso <- train_data_xform
train_Lasso$Gross <- NULL
fm.lasso <- glmnet(as.matrix(train_Lasso), as.double(train_data_xform$Gross),
    alpha = 1, standardize = TRUE, thresh = 1e-12)
plot(fm.lasso, xvar = "lambda")
```




```
print(fm.lasso$beta)
```

```
## 13 x 90 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 90 column names 's0', 's1', 's2' ... ]]

##
## Runtime             . .         .         .          .          .
## imdbRating          . .         .         .          .          .
## imdbVotes           . .         .         .         36.2662362 73.75953
## tomatoReviews       . .         .         .          .          .
## tomatoFresh         . .         .         .          .          .
## tomatoUserRating    . .         .         .          .          .
## Budget              . 0.2744273 0.5244752 0.7523096  0.9032544  1.03384
## imdbVotes_bin       . .         .         .          .          .
## Budget_bin          . .         .         .          .          .
## Runtime_log         . .         .         .          .          .
## imdbRating_P2       . .         .         .          .          .
## imdbVotes_P2        . .         .         .          .          .
## tomatoUserRating_P2 . .         .         .          .          .
##
## Runtime               .          .         .          .          .
## imdbRating            .          .         .          .          .
## imdbVotes           107.922041 139.04965 167.411961 193.254645 216.801537
## tomatoReviews         .          .         .          .          .
```

```
## tomatoFresh           .          .          .          .          .
## tomatoUserRating      .          .          .          .          .
## Budget                1.152825   1.26124    1.360024   1.450032   1.532043
## imdbVotes_bin         .          .          .          .          .
## Budget_bin            .          .          .          .          .
## Runtime_log           .          .          .          .          .
## imdbRating_P2         .          .          .          .          .
## imdbVotes_P2          .          .          .          .          .
## tomatoUserRating_P2   .          .          .          .          .
##
## Runtime               .          .          .          .          .
## imdbRating            .          .          .          .          .
## imdbVotes           238.256588 257.805631 275.617991 291.847950 306.636084
## tomatoReviews         .          .          .          .          .
## tomatoFresh           .          .          .          .          .
## tomatoUserRating      .          .          .          .          .
## Budget                1.606769   1.674857   1.736896   1.793424   1.844929
## imdbVotes_bin         .          .          .          .          .
## Budget_bin            .          .          .          .          .
## Runtime_log           .          .          .          .          .
## imdbRating_P2         .          .          .          .          .
## imdbVotes_P2          .          .          .          .          .
## tomatoUserRating_P2   .          .          .          .          .
##
## Runtime               .          .          .          .          .
## imdbRating            .          .          .          .          .
## imdbVotes           320.11048  332.387850 343.574531 353.767310 363.054699
## tomatoReviews         .          .          .          .          .
## tomatoFresh           .          .          .          .          .
## tomatoUserRating      .          .          .          .          .
## Budget                1.89186    1.934621   1.973583   2.009084   2.041431
## imdbVotes_bin         .          .          .          .          .
## Budget_bin            .          .          .          .          .
## Runtime_log           .          .          .          .          .
## imdbRating_P2         .          .          .          .          .
## imdbVotes_P2          .          .          .          .          .
## tomatoUserRating_P2   .          .          .          .          .
##
## Runtime               .            .            .          .
## imdbRating            .            .            .          .
## imdbVotes           371.517022   377.022688   381.264033 3.837091e+02
## tomatoReviews         .            .            .          .
## tomatoFresh           .        10811.796429 24463.701835 3.555444e+04
## tomatoUserRating      .            .            .          .
## Budget                2.070905     2.095691     2.117547 2.136114e+00
## imdbVotes_bin         .            .            .        9.145021e+05
## Budget_bin            .            .            .          .
## Runtime_log           .            .            .          .
## imdbRating_P2         .            .            .          .
## imdbVotes_P2          .            .            .          .
## tomatoUserRating_P2   .            .            .          .
##
## Runtime             .            .            .            .
## imdbRating          .            .            .            .
```

```
## imdbVotes           3.847462e+02 3.854873e+02 3.861623e+02 3.867774e+02
## tomatoReviews       .            .            .            .
## tomatoFresh         4.017896e+04 4.333453e+04 4.621039e+04 4.883077e+04
## tomatoUserRating    1.677130e+06 3.537955e+06 5.233429e+06 6.778282e+06
## Budget              2.157287e+00 2.177455e+00 2.195831e+00 2.212574e+00
## imdbVotes_bin       1.676481e+06 2.335720e+06 2.936425e+06 3.483766e+06
## Budget_bin          .            .            .            .
## Runtime_log         .            .            .            .
## imdbRating_P2       .            .            .            .
## imdbVotes_P2        .            .            .            .
## tomatoUserRating_P2 .            .            .            .
##
## Runtime             .            .            .            .
## imdbRating          .            .            .            .
## imdbVotes           3.873381e+02 3.878487e+02 3.883139e+02 3.887378e+02
## tomatoReviews       .            .            .            .
## tomatoFresh         5.121773e+04 5.339327e+04 5.537554e+04 5.718172e+04
## tomatoUserRating    8.185937e+06 9.468497e+06 1.063712e+07 1.170192e+07
## Budget              2.227831e+00 2.241731e+00 2.254397e+00 2.265938e+00
## imdbVotes_bin       3.982432e+06 4.436849e+06 4.850896e+06 5.228160e+06
## Budget_bin          .            .            .            .
## Runtime_log         .            .            .            .
## imdbRating_P2       .            .            .            .
## imdbVotes_P2        .            .            .            .
## tomatoUserRating_P2 .            .            .            .
##
## Runtime             .             .            .            .
## imdbRating          .             .            .            .
## imdbVotes           3.891241e+02  3.903974e+02  3.914317e+02  3.923740e+02
## tomatoReviews       .             .            .            .
## tomatoFresh         5.882744e+04  5.877416e+04  5.782088e+04  5.695190e+04
## tomatoUserRating    1.267213e+07  1.386214e+07  1.481004e+07  1.567374e+07
## Budget              2.276453e+00  2.307146e+00  2.349156e+00  2.387433e+00
## imdbVotes_bin       5.571909e+06  6.156745e+06  6.919622e+06  7.614768e+06
## Budget_bin          .            -2.024730e+06 -5.580260e+06 -8.819931e+06
## Runtime_log         .            -5.207248e+06 -1.016195e+07 -1.467647e+07
## imdbRating_P2       .             .            .            .
## imdbVotes_P2        .             .            .            .
## tomatoUserRating_P2 .             .            .            .
##
## Runtime              .             .             .             .
## imdbRating           .             .             .             .
## imdbVotes            3.932329e+02  3.940152e+02  3.947279e+02  3.953774e+02
## tomatoReviews        .             .             .             .
## tomatoFresh          5.616018e+04  5.543873e+04  5.478138e+04  5.418242e+04
## tomatoUserRating     1.646072e+07  1.717778e+07  1.783114e+07  1.842646e+07
## Budget               2.422310e+00  2.454088e+00  2.483044e+00  2.509427e+00
## imdbVotes_bin        8.248077e+06  8.825206e+06  9.351065e+06  9.830209e+06
## Budget_bin          -1.177175e+07 -1.446138e+07 -1.691208e+07 -1.914506e+07
## Runtime_log         -1.878993e+07 -2.253797e+07 -2.595303e+07 -2.906472e+07
## imdbRating_P2        .             .             .             .
## imdbVotes_P2         .             .             .             .
## tomatoUserRating_P2  .             .             .             .
##
```

```
## Runtime              .             .             .             .
## imdbRating           .             .             .             .
## imdbVotes            3.978872e+02  4.003239e+02  4.025441e+02  4.045671e+02
## tomatoReviews        .             .             .             .
## tomatoFresh          5.766399e+04  6.114941e+04  6.432517e+04  6.721880e+04
## tomatoUserRating     2.146246e+07  2.442310e+07  2.712074e+07  2.957874e+07
## Budget               2.526259e+00  2.541033e+00  2.554494e+00  2.566760e+00
## imdbVotes_bin        1.046297e+07  1.105469e+07  1.159383e+07  1.208508e+07
## Budget_bin          -2.119304e+07 -2.306007e+07 -2.476125e+07 -2.631129e+07
## Runtime_log         -3.052538e+07 -3.174918e+07 -3.286425e+07 -3.388027e+07
## imdbRating_P2       -1.353033e+05 -2.691221e+05 -3.910533e+05 -5.021526e+05
## imdbVotes_P2         .             .             .             .
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              .             .             .             .
## imdbRating           .             .             .             .
## imdbVotes            4.063979e+02  4.182942e+02  4.395652e+02  4.589457e+02
## tomatoReviews       -7.978327e+03 -4.095676e+04 -7.273505e+04 -1.016898e+05
## tomatoFresh          7.809873e+04  1.126338e+05  1.439173e+05  1.724213e+05
## tomatoUserRating     3.182336e+07  3.392424e+07  3.587966e+07  3.766135e+07
## Budget               2.579665e+00  2.595255e+00  2.608077e+00  2.619760e+00
## imdbVotes_bin        1.262614e+07  1.239642e+07  1.116638e+07  1.004570e+07
## Budget_bin          -2.762251e+07 -2.853651e+07 -2.939379e+07 -3.017493e+07
## Runtime_log         -3.475123e+07 -3.536445e+07 -3.590871e+07 -3.640464e+07
## imdbRating_P2       -6.138108e+05 -7.534343e+05 -8.872799e+05 -1.009233e+06
## imdbVotes_P2         .            -9.550251e-06 -2.794652e-05 -4.470785e-05
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              .             .             .             .
## imdbRating           .             .             .             .
## imdbVotes            4.766046e+02  4.926946e+02  5.073552e+02  5.207144e+02
## tomatoReviews       -1.280724e+05 -1.521111e+05 -1.740143e+05 -1.939723e+05
## tomatoFresh          1.983930e+05  2.220575e+05  2.436197e+05  2.632670e+05
## tomatoUserRating     3.928475e+07  4.076394e+07  4.211171e+07  4.333979e+07
## Budget               2.630406e+00  2.640105e+00  2.648943e+00  2.656995e+00
## imdbVotes_bin        9.024576e+06  8.094168e+06  7.246417e+06  6.473885e+06
## Budget_bin          -3.088667e+07 -3.153519e+07 -3.212609e+07 -3.266448e+07
## Runtime_log         -3.685651e+07 -3.726823e+07 -3.764339e+07 -3.798518e+07
## imdbRating_P2       -1.120353e+06 -1.221600e+06 -1.313853e+06 -1.397914e+06
## imdbVotes_P2        -5.998014e-05 -7.389566e-05 -8.657496e-05 -9.812862e-05
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              .             .             .             .
## imdbRating           .             .             .             .
## imdbVotes            5.328859e+02  5.439761e+02  5.540810e+02  5.632883e+02
## tomatoReviews       -2.121568e+05 -2.287257e+05 -2.438227e+05 -2.575785e+05
## tomatoFresh          2.811683e+05  2.974792e+05  3.123412e+05  3.258828e+05
## tomatoUserRating     4.445874e+07  4.547828e+07  4.640725e+07  4.725369e+07
## Budget               2.664333e+00  2.671018e+00  2.677110e+00  2.682661e+00
## imdbVotes_bin        5.770069e+06  5.128780e+06  4.544462e+06  4.012055e+06
## Budget_bin          -3.315505e+07 -3.360205e+07 -3.400934e+07 -3.438045e+07
## Runtime_log         -3.829664e+07 -3.858043e+07 -3.883901e+07 -3.907462e+07
## imdbRating_P2       -1.474504e+06 -1.544290e+06 -1.607876e+06 -1.665814e+06
## imdbVotes_P2        -1.086552e-04 -1.182466e-04 -1.269859e-04 -1.349488e-04
```

```
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              .             .             .             .
## imdbRating           .             .             .            5.374125e+06
## imdbVotes            5.716775e+02  5.793225e+02  5.862874e+02  5.959706e+02
## tomatoReviews       -2.701122e+05 -2.815332e+05 -2.919389e+05 -3.061860e+05
## tomatoFresh          3.382214e+05  3.494645e+05  3.597083e+05  3.734558e+05
## tomatoUserRating     4.802494e+07  4.872770e+07  4.936800e+07  5.014061e+07
## Budget               2.687718e+00  2.692326e+00  2.696524e+00  2.698114e+00
## imdbVotes_bin        3.526948e+06  3.084843e+06  2.682100e+06  2.252710e+06
## Budget_bin          -3.471858e+07 -3.502666e+07 -3.530739e+07 -3.569093e+07
## Runtime_log         -3.928930e+07 -3.948487e+07 -3.966310e+07 -3.946107e+07
## imdbRating_P2       -1.718604e+06 -1.766707e+06 -1.810535e+06 -2.311977e+06
## imdbVotes_P2        -1.422043e-04 -1.488160e-04 -1.548396e-04 -1.606612e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              .            8.742070e+04  2.021975e+05  3.067772e+05
## imdbRating           1.172058e+07  1.779417e+07  2.344547e+07  2.859469e+07
## imdbVotes            6.056912e+02  6.148833e+02  6.233949e+02  6.311502e+02
## tomatoReviews       -3.204570e+05 -3.328456e+05 -3.438895e+05 -3.539522e+05
## tomatoFresh          3.871778e+05  3.997516e+05  4.112396e+05  4.217069e+05
## tomatoUserRating     5.089660e+07  5.158852e+07  5.222048e+07  5.279629e+07
## Budget               2.698960e+00  2.698799e+00  2.698276e+00  2.697800e+00
## imdbVotes_bin        1.845195e+06  1.470260e+06  1.127143e+06  8.145110e+05
## Budget_bin          -3.607497e+07 -3.641098e+07 -3.671154e+07 -3.698540e+07
## Runtime_log         -3.917815e+07 -4.887745e+07 -6.171532e+07 -7.341263e+07
## imdbRating_P2       -2.893413e+06 -3.450007e+06 -3.967971e+06 -4.439917e+06
## imdbVotes_P2        -1.660534e-04 -1.714134e-04 -1.764777e-04 -1.810921e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              4.020662e+05  4.888902e+05  5.680008e+05  6.400485e+05
## imdbRating           3.328646e+07  3.756173e+07  4.145691e+07  4.499708e+07
## imdbVotes            6.382166e+02  6.446560e+02  6.505226e+02  6.550723e+02
## tomatoReviews       -3.631210e+05 -3.714768e+05 -3.790889e+05 -3.860785e+05
## tomatoFresh          4.312443e+05  4.399361e+05  4.478542e+05  4.551027e+05
## tomatoUserRating     5.332094e+07  5.379915e+07  5.423473e+07  5.462637e+07
## Budget               2.697367e+00  2.696972e+00  2.696612e+00  2.696317e+00
## imdbVotes_bin        5.296525e+05  2.700874e+05  3.359302e+04  .
## Budget_bin          -3.723493e+07 -3.746229e+07 -3.766945e+07 -3.787047e+07
## Runtime_log         -8.407077e+07 -9.378200e+07 -1.026306e+08 -1.106826e+08
## imdbRating_P2       -4.869936e+06 -5.261787e+06 -5.618796e+06 -5.943552e+06
## imdbVotes_P2        -1.852966e-04 -1.891281e-04 -1.926187e-04 -1.952230e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              7.056921e+05  7.655037e+05  8.200018e+05  8.696584e+05
## imdbRating           4.822174e+07  5.115981e+07  5.383686e+07  5.627609e+07
## imdbVotes            6.590848e+02  6.627406e+02  6.660717e+02  6.691069e+02
## tomatoReviews       -3.924581e+05 -3.982707e+05 -4.035669e+05 -4.083926e+05
## tomatoFresh          4.617148e+05  4.677393e+05  4.732286e+05  4.782302e+05
## tomatoUserRating     5.498253e+07  5.530704e+07  5.560271e+07  5.587212e+07
## Budget               2.696054e+00  2.695815e+00  2.695597e+00  2.695398e+00
## imdbVotes_bin        .             .             .             .
## Budget_bin          -3.805569e+07 -3.822445e+07 -3.837822e+07 -3.851833e+07
## Runtime_log         -1.180177e+08 -1.247011e+08 -1.307907e+08 -1.363394e+08
```

```
## imdbRating_P2       -6.239418e+06 -6.508988e+06 -6.754610e+06 -6.978412e+06
## imdbVotes_P2        -1.974994e-04 -1.995736e-04 -2.014634e-04 -2.031854e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              9.149041e+05  9.561299e+05  9.936933e+05  1.027920e+06
## imdbRating           5.849876e+07  6.052384e+07  6.236902e+07  6.405027e+07
## imdbVotes            6.718725e+02  6.743923e+02  6.766883e+02  6.787803e+02
## tomatoReviews       -4.127899e+05 -4.167963e+05 -4.204467e+05 -4.237729e+05
## tomatoFresh          4.827878e+05  4.869402e+05  4.907237e+05  4.941711e+05
## tomatoUserRating     5.611761e+07  5.634128e+07  5.654507e+07  5.673076e+07
## Budget               2.695217e+00  2.695052e+00  2.694901e+00  2.694764e+00
## imdbVotes_bin        .             .             .             .
## Budget_bin          -3.864600e+07 -3.876232e+07 -3.886831e+07 -3.896489e+07
## Runtime_log         -1.413952e+08 -1.460018e+08 -1.501992e+08 -1.540237e+08
## imdbRating_P2       -7.182344e+06 -7.368148e+06 -7.537444e+06 -7.691701e+06
## imdbVotes_P2        -2.047545e-04 -2.061841e-04 -2.074867e-04 -2.086736e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              1.059119e+06  1.087572e+06  1.113498e+06  1.137120e+06
## imdbRating           6.558537e+07  6.699106e+07  6.827180e+07  6.943906e+07
## imdbVotes            6.809694e+02  6.836101e+02  6.860160e+02  6.882091e+02
## tomatoReviews       -4.267839e+05 -4.294826e+05 -4.319415e+05 -4.341835e+05
## tomatoFresh          4.972998e+05  5.001219e+05  5.026931e+05  5.050375e+05
## tomatoUserRating     5.690177e+07  5.706172e+07  5.720744e+07  5.734038e+07
## Budget               2.694628e+00  2.694476e+00  2.694338e+00  2.694211e+00
## imdbVotes_bin       -6.472834e+04 -2.714808e+05 -4.598626e+05 -6.315217e+05
## Budget_bin          -3.904852e+07 -3.911477e+07 -3.917513e+07 -3.923013e+07
## Runtime_log         -1.575123e+08 -1.606993e+08 -1.636032e+08 -1.662489e+08
## imdbRating_P2       -7.832444e+06 -7.961086e+06 -8.078294e+06 -8.185123e+06
## imdbVotes_P2        -2.099599e-04 -2.115999e-04 -2.130941e-04 -2.144561e-04
## tomatoUserRating_P2  .             .             .             .
##
## Runtime              1.158644e+06  1.178256e+06  1.196404e+06  1.212857e+06
## imdbRating           7.050235e+07  7.147100e+07  7.203119e+07  7.235418e+07
## imdbVotes            6.902065e+02  6.920260e+02  6.937158e+02  6.952287e+02
## tomatoReviews       -4.362249e+05 -4.380835e+05 -4.399714e+05 -4.417259e+05
## tomatoFresh          5.071722e+05  5.091158e+05  5.111845e+05  5.131478e+05
## tomatoUserRating     5.746137e+07  5.757146e+07  5.950458e+07  6.210555e+07
## Budget               2.694097e+00  2.693992e+00  2.694075e+00  2.694258e+00
## imdbVotes_bin       -7.879194e+05 -9.304165e+05 -1.075774e+06 -1.213866e+06
## Budget_bin          -3.928025e+07 -3.932591e+07 -3.937400e+07 -3.942022e+07
## Runtime_log         -1.686598e+08 -1.708565e+08 -1.729008e+08 -1.747633e+08
## imdbRating_P2       -8.282431e+06 -8.371072e+06 -8.426021e+06 -8.460696e+06
## imdbVotes_P2        -2.156967e-04 -2.168266e-04 -2.178564e-04 -2.187778e-04
## tomatoUserRating_P2  .             .            -2.816603e+05 -6.679156e+05
##
## Runtime              1.227853e+06  1.241518e+06  1.253968e+06  1.265313e+06
## imdbRating           7.265001e+07  7.291958e+07  7.316521e+07  7.338901e+07
## imdbVotes            6.966076e+02  6.978640e+02  6.990088e+02  7.000518e+02
## tomatoReviews       -4.433246e+05 -4.447814e+05 -4.461087e+05 -4.473180e+05
## tomatoFresh          5.149367e+05  5.165666e+05  5.180517e+05  5.194049e+05
## tomatoUserRating     6.447040e+07  6.662509e+07  6.858829e+07  7.037701e+07
## Budget               2.694424e+00  2.694575e+00  2.694713e+00  2.694839e+00
## imdbVotes_bin       -1.339662e+06 -1.454283e+06 -1.558719e+06 -1.653877e+06
```

```
## Budget_bin          -3.946233e+07 -3.950070e+07 -3.953565e+07 -3.956751e+07
## Runtime_log         -1.764609e+08 -1.780077e+08 -1.794170e+08 -1.807012e+08
## imdbRating_P2       -8.492419e+06 -8.521324e+06 -8.547662e+06 -8.571660e+06
## imdbVotes_P2        -2.196175e-04 -2.203826e-04 -2.210797e-04 -2.217149e-04
## tomatoUserRating_P2 -1.019079e+06 -1.339033e+06 -1.630552e+06 -1.896164e+06
##
## Runtime              1.275650e+06
## imdbRating           7.359307e+07
## imdbVotes            7.010023e+02
## tomatoReviews       -4.484200e+05
## tomatoFresh          5.206379e+05
## tomatoUserRating     7.200653e+07
## Budget               2.694953e+00
## imdbVotes_bin       -1.740581e+06
## Budget_bin          -3.959653e+07
## Runtime_log         -1.818713e+08
## imdbRating_P2       -8.593539e+06
## imdbVotes_P2        -2.222936e-04
## tomatoUserRating_P2 -2.138132e+06
```

```
fm = lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
    Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
    Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh + imdbVotes:tomatoReviews +
    imdbVotes:tomatoFresh + imdbVotes:Budget + tomatoReviews:imdbRating_P2 +
    tomatoReviews:tomatoUserRating_P2 + tomatoUserRating:Runtime_log +
    Budget:imdbVotes_bin + Budget:Runtime_log + Runtime_log:imdbVotes_P2 +
    Runtime_log:tomatoUserRating_P2, data = train_data_xform)

confint(fm, "(Intercept)", level = 0.95)
```

```
##                     2.5 %      97.5 %
## (Intercept) -42950580803 -1652736647
```

```
confint(fm, "Year", level = 0.95)
```

```
##      2.5 % 97.5 %
## Year    NA     NA
```

```
confint(fm, "Runtime", level = 0.95)
```

```
##              2.5 %    97.5 %
## Runtime -122367765 -18086123
```

```
confint(fm, "imdbRating", level = 0.95)
```

```
##               2.5 %   97.5 %
## imdbRating -4712561 62504631
```

```
confint(fm, "imdbVotes", level = 0.95)
```

```
##             2.5 %   97.5 %
## imdbVotes 640.916 843.7373
```

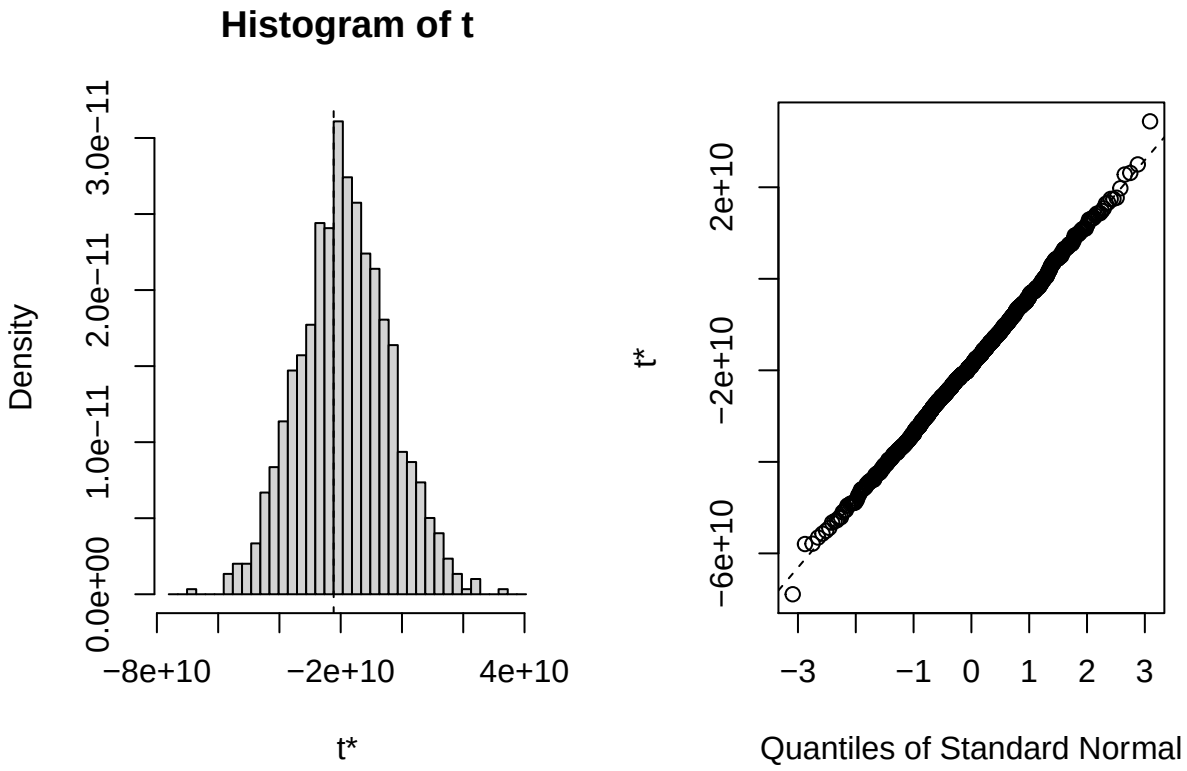
```
confint(fm, "tomatoMeter", level = 0.95)
```

```
##             2.5 % 97.5 %
## tomatoMeter    NA     NA
```

```
mean(lm.bootstrap$t) - lm.bootstrap$t0
```

```
##                         (Intercept)                             Runtime
##                          22090567287                           -140864493
##                           imdbRating                           imdbVotes
##                            -239987472                           -211092180
##                          tomatoReviews                         tomatoFresh
##                             -210900784                           -211722556
##                       tomatoUserRating                              Budget
##                          -13030621333                           -211091474
##                         imdbVotes_bin                           Budget_bin
##                             -205522650                           -326954142
##                          Runtime_log                         imdbRating_P2
##                           -6543153557                           -209521331
##                         imdbVotes_P2                    tomatoUserRating_P2
##                             -211091438                           1634071453
##                  Runtime:tomatoFresh              Runtime:tomatoUserRating
##                             -211096196                           -252702062
##                       Runtime:Budget                    Runtime:Budget_bin
##                             -211091438                           -210098030
##                  Runtime:imdbVotes_P2            Runtime:tomatoUserRating_P2
##                             -211091437                           -204972687
##                imdbRating:tomatoFresh                imdbVotes:tomatoReviews
##                             -210982661                           -211091436
##                 imdbVotes:tomatoFresh                       imdbVotes:Budget
##                             -211091438                           -211091437
##       tomatoReviews:imdbRating_P2 tomatoReviews:tomatoUserRating_P2
##                             -211055156                           -211228435
##      tomatoUserRating:Runtime_log                  Budget:imdbVotes_bin
##                             3464640313                           -211091438
##                    Budget:Runtime_log               Runtime_log:imdbVotes_P2
##                             -211091428                           -211091437
##   Runtime_log:tomatoUserRating_P2
##                             -741992657
```

```
plot(lm.bootstrap)
```



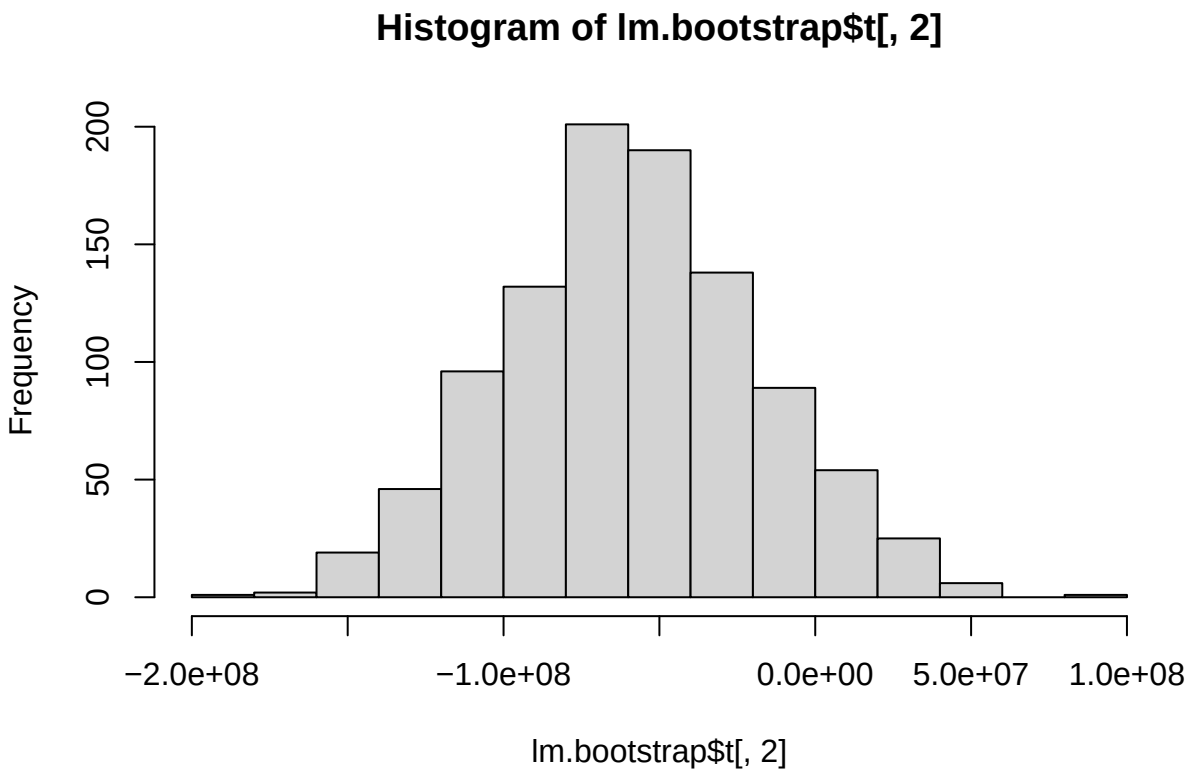

```
percentile_ci <- boot.ci(lm.bootstrap, type = "perc")
print(percentile_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   (-47396881173,  10958749402 )
## Calculations and Intervals on Original Scale
```

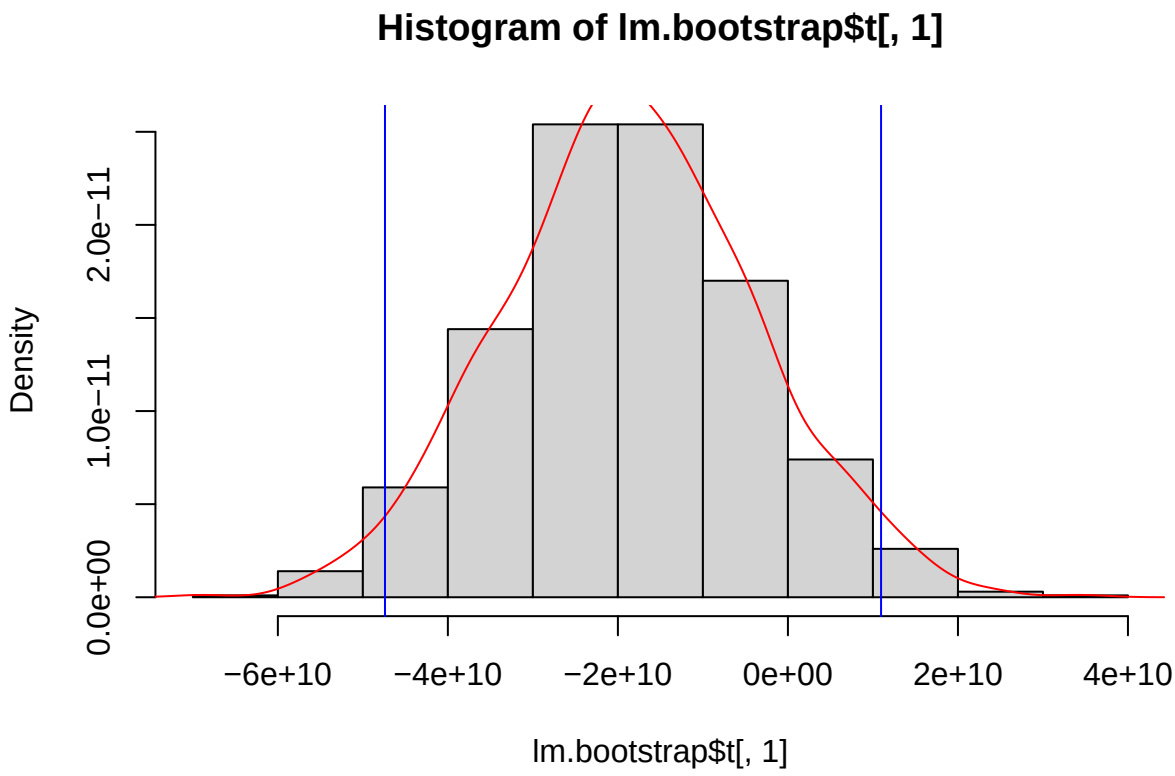
```
ci_H = percentile_ci$perc[, c(4, 5)]
print(ci_H)
```

```
##
## -47396881173  10958749402
```

```
hist(lm.bootstrap$t[, 2])
```

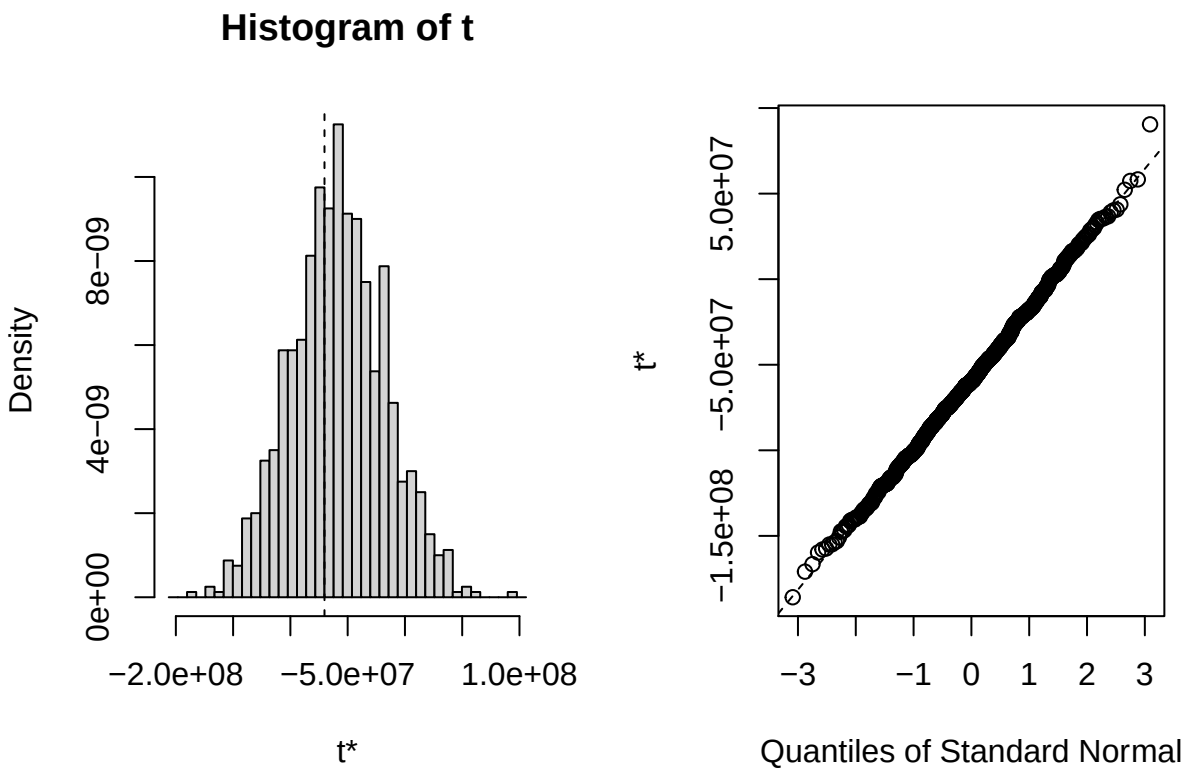



```
hist(lm.bootstrap$t[, 1], probability = T)
lines(density(lm.bootstrap$t[, 1]), col = "red")
abline(v = ci_H, col = "blue")
```




```
# Plotting bootstrap results for 'Year' Covariate

plot(lm.bootstrap, index = 2)
```




```
### Another method for Bootstrapping

# selected.cov<-c('Gross','Year','Runtime','imdbRating','imdbVotes','tomatoMeter','tomatoRating','tomat


X <- train_data_xform
X$Gross <- NULL
Y <- train_data_xform$Gross
df = data.frame(X, Y)

coef.boot = function(data, indices) {
    fm = lm(Gross ~ . + Runtime:tomatoFresh + Runtime:tomatoUserRating +
        Runtime:Budget + Runtime:Budget_bin + Runtime:imdbVotes_P2 +
        Runtime:tomatoUserRating_P2 + imdbRating:tomatoFresh +
        imdbVotes:tomatoReviews + imdbVotes:tomatoFresh + imdbVotes:Budget +
        tomatoReviews:imdbRating_P2 + tomatoReviews:tomatoUserRating_P2 +
        tomatoUserRating:Runtime_log + Budget:imdbVotes_bin +
        Budget:Runtime_log + Runtime_log:imdbVotes_P2 + Runtime_log:tomatoUserRating_P2,
        data = train_data_xform)
    return(coef(fm))
}

boot.out = boot(df, coef.boot, 1000)
summary(boot.out)
```

```
##           Length Class      Mode
```

```
## t0           31  -none-     numeric
## t         31000  -none-     numeric
## R             1  -none-     numeric
## data         14  data.frame list
## seed        626  -none-     numeric
## statistic     1  -none-     function
## sim           1  -none-     character
## call          4  -none-     call
## stype         1  -none-     character
## strata     3277  -none-     numeric
## weights    3277  -none-     numeric
```

```
# Getting Confidence intervals for Bootstrap estimates
for (i in 1:31) {
    print(i)
    print(boot.ci(lm.bootstrap, type = "perc", index = i))

}
```

```
## [1] 1
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-47396881173,  10958749402 )
## Calculations and Intervals on Original Scale
## [1] 2
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-138925637,   24026161 )
## Calculations and Intervals on Original Scale
## [1] 3
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 1281864, 56435557 )
## Calculations and Intervals on Original Scale
## [1] 4
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
```

```
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (537.5, 953.0 )
## Calculations and Intervals on Original Scale
## [1] 5
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-659022,  258604 )
## Calculations and Intervals on Original Scale
## [1] 6
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-860652, 2184220 )
## Calculations and Intervals on Original Scale
## [1] 7
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-6120316193, 28082254177 )
## Calculations and Intervals on Original Scale
## [1] 8
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 3.45, 70.85 )
## Calculations and Intervals on Original Scale
## [1] 9
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-29309313,  17958726 )
## Calculations and Intervals on Original Scale
## [1] 10
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 31689668, 193617528 )
## Calculations and Intervals on Original Scale
## [1] 11
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-2965128484, 13302220589 )
## Calculations and Intervals on Original Scale
## [1] 12
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-4203727,  1266468 )
## Calculations and Intervals on Original Scale
## [1] 13
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0031,  0.0320 )
## Calculations and Intervals on Original Scale
```

```
## [1] 14
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-4055104093,   849718479 )
## Calculations and Intervals on Original Scale
## [1] 15
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( -998, 10283 )
## Calculations and Intervals on Original Scale
## [1] 16
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-10205645,  80726275 )
## Calculations and Intervals on Original Scale
## [1] 17
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-0.0001,  0.1565 )
## Calculations and Intervals on Original Scale
## [1] 18
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-1721423,  -259709 )
```

```
## Calculations and Intervals on Original Scale
## [1] 19
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 0.0000,  0.0001 )
## Calculations and Intervals on Original Scale
## [1] 20
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-11696156,   1162651 )
## Calculations and Intervals on Original Scale
## [1] 21
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-351111,  127355 )
## Calculations and Intervals on Original Scale
## [1] 22
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-3.371,  0.315 )
## Calculations and Intervals on Original Scale
## [1] 23
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
```

```
## 95%   (-1.2843,  2.6900 )
## Calculations and Intervals on Original Scale
## [1] 24
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 0,  0 )
## Calculations and Intervals on Original Scale
## [1] 25
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-49696, -22292 )
## Calculations and Intervals on Original Scale
## [1] 26
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 93109, 180321 )
## Calculations and Intervals on Original Scale
## [1] 27
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-7809800792,  1588080816 )
## Calculations and Intervals on Original Scale
## [1] 28
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
```

```
## Level     Percentile
## 95%   (-0.1339,  0.7285 )
## Calculations and Intervals on Original Scale
## [1] 29
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-18.639,  -0.640 )
## Calculations and Intervals on Original Scale
## [1] 30
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-0.0083, -0.0010 )
## Calculations and Intervals on Original Scale
## [1] 31
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = lm.bootstrap, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-214350096, 1132488709 )
## Calculations and Intervals on Original Scale
```

```r
# Getting Confidence intervals for Regression estimates
conf_int_OLS <- confint(finalModel, level = 0.95)
print(conf_int_OLS)
```

```
##                                       2.5 %        97.5 %
## (Intercept)                   -4.295058e+10 -1.652737e+09
## Runtime                       -1.223678e+08 -1.808612e+07
## imdbRating                    -4.712561e+06  6.250463e+07
## imdbVotes                      6.409160e+02  8.437373e+02
## tomatoReviews                 -5.446368e+05  1.633295e+05
## tomatoFresh                   -5.139741e+05  1.776211e+06
## tomatoUserRating               1.157672e+09  2.448139e+10
## Budget                         2.244065e+01  5.009037e+01
## imdbVotes_bin                 -2.059363e+07  9.456052e+06
## Budget_bin                     6.168875e+07  1.700367e+08
## Runtime_log                    7.180900e+08  1.194603e+10
## imdbRating_P2                 -4.605517e+06  1.465305e+06
```

```
## imdbVotes_P2                         1.272356e-02  2.406328e-02
## tomatoUserRating_P2                 -3.481731e+09 -2.085951e+08
## Runtime:tomatoFresh                  1.239913e+03  8.277776e+03
## Runtime:tomatoUserRating             1.244742e+07  7.077382e+07
## Runtime:Budget                       4.358390e-02  1.026539e-01
## Runtime:Budget_bin                  -1.464122e+06 -5.226927e+05
## Runtime:imdbVotes_P2                 2.653743e-05  4.521615e-05
## Runtime:tomatoUserRating_P2         -1.017944e+07 -2.058065e+06
## imdbRating:tomatoFresh              -2.883923e+05  7.083996e+04
## imdbVotes:tomatoReviews             -2.563888e+00 -7.440296e-01
## imdbVotes:tomatoFresh               -3.111550e-01  1.761843e+00
## imdbVotes:Budget                     4.829653e-06  5.997546e-06
## tomatoReviews:imdbRating_P2         -4.632763e+04 -2.623448e+04
## tomatoReviews:tomatoUserRating_P2    1.109310e+05  1.630632e+05
## tomatoUserRating:Runtime_log        -6.838810e+09 -5.126535e+08
## Budget:imdbVotes_bin                 5.517613e-02  5.243142e-01
## Budget:Runtime_log                  -1.290942e+01 -5.651354e+00
## Runtime_log:imdbVotes_P2            -6.245568e-03 -3.438912e-03
## Runtime_log:tomatoUserRating_P2      8.779301e+07  9.740094e+08
```

```
# Extracting p-values from the final model
p.vals <- summary(finalModel)$coef[, 4]
p.vals
```

```
##                         (Intercept)                             Runtime
##                        3.428235e-02                        8.310616e-03
##                           imdbRating                            imdbVotes
##                        9.193650e-02                        2.445510e-45
##                         tomatoReviews                         tomatoFresh
##                        2.910372e-01                        2.799384e-01
##                     tomatoUserRating                              Budget
##                        3.120908e-02                        2.859387e-07
##                        imdbVotes_bin                          Budget_bin
##                        4.674563e-01                        2.821871e-05
##                         Runtime_log                       imdbRating_P2
##                        2.707135e-02                        3.105652e-01
##                        imdbVotes_P2                  tomatoUserRating_P2
##                        2.290855e-10                        2.713283e-02
##                  Runtime:tomatoFresh            Runtime:tomatoUserRating
##                        8.051146e-03                        5.179406e-03
##                     Runtime:Budget                   Runtime:Budget_bin
##                        1.266844e-06                        3.594059e-05
##                Runtime:imdbVotes_P2         Runtime:tomatoUserRating_P2
##                        6.442507e-14                        3.155004e-03
##              imdbRating:tomatoFresh             imdbVotes:tomatoReviews
##                        2.351552e-01                        3.706292e-04
##               imdbVotes:tomatoFresh                     imdbVotes:Budget
##                        1.701276e-01                        2.171711e-70
##       tomatoReviews:imdbRating_P2 tomatoReviews:tomatoUserRating_P2
##                        1.751317e-12                        1.591478e-24
##      tomatoUserRating:Runtime_log                Budget:imdbVotes_bin
##                        2.276309e-02                        1.549419e-02
##                  Budget:Runtime_log            Runtime_log:imdbVotes_P2
##                        5.616571e-07                        1.570070e-11
```

```
##   Runtime_log:tomatoUserRating_P2
##                      1.887507e-02
```

```
# Using Benjamin-Hochberg to adjust p-values for multiple
# hypothesis testing
benj.hoch <- p.adjust(p.vals, method = "BH", n = 31)
benj.hoch
```

```
##                        (Intercept)                            Runtime
##                       4.428136e-02                       1.515465e-02
##                          imdbRating                            imdbVotes
##                       1.140013e-01                       3.790540e-44
##                      tomatoReviews                         tomatoFresh
##                       3.111087e-01                       3.099318e-01
##                    tomatoUserRating                             Budget
##                       4.206441e-02                       1.108012e-06
##                      imdbVotes_bin                          Budget_bin
##                       4.674563e-01                       7.952545e-05
##                        Runtime_log                      imdbRating_P2
##                       3.823262e-02                       3.209173e-01
##                      imdbVotes_P2                 tomatoUserRating_P2
##                       1.014522e-09                       3.823262e-02
##                 Runtime:tomatoFresh            Runtime:tomatoUserRating
##                       1.515465e-02                       1.070411e-02
##                     Runtime:Budget                  Runtime:Budget_bin
##                       3.927215e-06                       9.284653e-05
##                Runtime:imdbVotes_P2         Runtime:tomatoUserRating_P2
##                       4.992943e-13                       6.986080e-03
##              imdbRating:tomatoFresh             imdbVotes:tomatoReviews
##                       2.699930e-01                       8.838082e-04
##               imdbVotes:tomatoFresh                    imdbVotes:Budget
##                       2.028445e-01                       6.732303e-69
##        tomatoReviews:imdbRating_P2 tomatoReviews:tomatoUserRating_P2
##                       1.085817e-11                       1.644527e-23
##       tomatoUserRating:Runtime_log                Budget:imdbVotes_bin
##                       3.528279e-02                       2.668444e-02
##                  Budget:Runtime_log           Runtime_log:imdbVotes_P2
##                       1.934597e-06                       8.112028e-11
##   Runtime_log:tomatoUserRating_P2
##                       3.079617e-02
```

```
# Using Bonferroni to adjust p-values for multiple hypothesis
# testing
bonferroni <- p.adjust(p.vals, method = "bonferroni", n = 31)
bonferroni
```

```
##                        (Intercept)                            Runtime
##                       1.000000e+00                       2.576291e-01
##                          imdbRating                            imdbVotes
##                       1.000000e+00                       7.581080e-44
##                      tomatoReviews                         tomatoFresh
##                       1.000000e+00                       1.000000e+00
##                    tomatoUserRating                             Budget
```

```
##                      9.674815e-01                      8.864099e-06
##                     imdbVotes_bin                        Budget_bin
##                      1.000000e+00                      8.747800e-04
##                       Runtime_log                     imdbRating_P2
##                      8.392119e-01                      1.000000e+00
##                      imdbVotes_P2                tomatoUserRating_P2
##                      7.101651e-09                      8.411177e-01
##                Runtime:tomatoFresh           Runtime:tomatoUserRating
##                      2.495855e-01                      1.605616e-01
##                     Runtime:Budget                 Runtime:Budget_bin
##                      3.927215e-05                      1.114158e-03
##                Runtime:imdbVotes_P2        Runtime:tomatoUserRating_P2
##                      1.997177e-12                      9.780512e-02
##             imdbRating:tomatoFresh            imdbVotes:tomatoReviews
##                      1.000000e+00                      1.148951e-02
##              imdbVotes:tomatoFresh                   imdbVotes:Budget
##                      1.000000e+00                      6.732303e-69
##       tomatoReviews:imdbRating_P2 tomatoReviews:tomatoUserRating_P2
##                      5.429083e-11                      4.933582e-23
##      tomatoUserRating:Runtime_log               Budget:imdbVotes_bin
##                      7.056558e-01                      4.803199e-01
##                 Budget:Runtime_log           Runtime_log:imdbVotes_P2
##                      1.741137e-05                      4.867217e-10
##   Runtime_log:tomatoUserRating_P2
##                      5.851272e-01
```

```
# Comparing p-values from all three methods
p.val_table <- data.frame(p.vals, benj.hoch, bonferroni)
colnames(p.val_table) <- c("OLS", "Benjamini-Hochberg", "Bonferroni")
kable(p.val_table, caption = "P-value comparison for all three models",
    align = "c")
```

```
# Creating Regression summary for Final Model using jtools
library(jtools)
summ(finalModel, confint = TRUE, digits = 3)
```

```
# Creating regression summary table of our Final model on
# Test set

summ(finModel.test, confint = TRUE, digits = 3)
```

```
# Using R to generate Latex Code for Regression output
# library(stargazer) stargazer(finalModel, title = 'OLS
# Results', align = TRUE, type = 'latex', ci = TRUE)
```

Table 1: P-value comparison for all three models

	OLS	Benjamini-Hochberg	Bonferroni
(Intercept)	0.0342823	0.0442814	1.0000000
Runtime	0.0083106	0.0151547	0.2576291
imdbRating	0.0919365	0.1140013	1.0000000
imdbVotes	0.0000000	0.0000000	0.0000000
tomatoReviews	0.2910372	0.3111087	1.0000000
tomatoFresh	0.2799384	0.3099318	1.0000000
tomatoUserRating	0.0312091	0.0420644	0.9674815
Budget	0.0000003	0.0000011	0.0000089
imdbVotes_bin	0.4674563	0.4674563	1.0000000
Budget_bin	0.0000282	0.0000795	0.0008748
Runtime_log	0.0270714	0.0382326	0.8392119
imdbRating_P2	0.3105652	0.3209173	1.0000000
imdbVotes_P2	0.0000000	0.0000000	0.0000000
tomatoUserRating_P2	0.0271328	0.0382326	0.8411177
Runtime:tomatoFresh	0.0080511	0.0151547	0.2495855
Runtime:tomatoUserRating	0.0051794	0.0107041	0.1605616
Runtime:Budget	0.0000013	0.0000039	0.0000393
Runtime:Budget_bin	0.0000359	0.0000928	0.0011142
Runtime:imdbVotes_P2	0.0000000	0.0000000	0.0000000
Runtime:tomatoUserRating_P2	0.0031550	0.0069861	0.0978051
imdbRating:tomatoFresh	0.2351552	0.2699930	1.0000000
imdbVotes:tomatoReviews	0.0003706	0.0008838	0.0114895
imdbVotes:tomatoFresh	0.1701276	0.2028445	1.0000000
imdbVotes:Budget	0.0000000	0.0000000	0.0000000
tomatoReviews:imdbRating_P2	0.0000000	0.0000000	0.0000000
tomatoReviews:tomatoUserRating_P2	0.0000000	0.0000000	0.0000000
tomatoUserRating:Runtime_log	0.0227631	0.0352828	0.7056558
Budget:imdbVotes_bin	0.0154942	0.0266844	0.4803199
Budget:Runtime_log	0.0000006	0.0000019	0.0000174
Runtime_log:imdbVotes_P2	0.0000000	0.0000000	0.0000000
Runtime_log:tomatoUserRating_P2	0.0188751	0.0307962	0.5851272

Observations	3277
Dependent variable	Gross
Type	OLS linear regression

F(30,3246)	342.141
R^2	0.760
Adj. R^2	0.758

	Est.	2.5%	97.5%	t val.	p
(Intercept)	-22301658724.955	-42950580803.268	-1652736646.642	-2.118	0.034
Runtime	-70226944.019	-122367764.663	-18086123.376	-2.641	0.008
imdbRating	28896035.007	-4712560.688	62504630.701	1.686	0.092
imdbVotes	742.327	640.916	843.737	14.352	0.000
tomatoReviews	-190653.652	-544636.823	163329.519	-1.056	0.291
tomatoFresh	631118.351	-513974.088	1776210.791	1.081	0.280
tomatoUserRating	12819529895.951	1157671504.432	24481388287.469	2.155	0.031
Budget	36.266	22.441	50.090	5.143	0.000
imdbVotes_bin	-5568787.500	-20593626.754	9456051.754	-0.727	0.467
Budget_bin	115862704.818	61688747.148	170036662.488	4.193	0.000
Runtime_log	6332062119.545	718090042.420	11946034196.669	2.211	0.027
imdbRating_P2	-1570106.231	-4605517.280	1465304.817	-1.014	0.311
imdbVotes_P2	0.018	0.013	0.024	6.361	0.000
tomatoUserRating_P2	-1845162890.097	-3481730729.298	-208595050.895	-2.211	0.027
Runtime:tomatoFresh	4758.844	1239.913	8277.776	2.652	0.008
Runtime:tomatoUserRating	41610624.598	12447424.230	70773824.965	2.798	0.005
Runtime:Budget	0.073	0.044	0.103	4.854	0.000
Runtime:Budget_bin	-993407.519	-1464122.359	-522692.679	-4.138	0.000
Runtime:imdbVotes_P2	0.000	0.000	0.000	7.532	0.000
Runtime:tomatoUserRating_P2	-6118750.647	-10179436.168	-2058065.126	-2.954	0.003
imdbRating:tomatoFresh	-108776.174	-288392.308	70839.960	-1.187	0.235
imdbVotes:tomatoReviews	-1.654	-2.564	-0.744	-3.564	0.000
imdbVotes:tomatoFresh	0.725	-0.311	1.762	1.372	0.170
imdbVotes:Budget	0.000	0.000	0.000	18.177	0.000
tomatoReviews:imdbRating_P2	-36281.058	-46327.632	-26234.485	-7.081	0.000
tomatoReviews:tomatoUserRating_P2	136997.091	110930.975	163063.208	10.305	0.000
tomatoUserRating:Runtime_log	-3675731750.037	-6838810035.321	-512653464.753	-2.278	0.023
Budget:imdbVotes_bin	0.290	0.055	0.524	2.422	0.015
Budget:Runtime_log	-9.280	-12.909	-5.651	-5.014	0.000
Runtime_log:imdbVotes_P2	-0.005	-0.006	-0.003	-6.765	0.000
Runtime_log:tomatoUserRating_P2	530901219.067	87793014.911	974009423.223	2.349	0.019

Standard errors: OLS

Observations	816
Dependent variable	Gross
Type	OLS linear regression

F(30,785)	76.557
R^2	0.745
Adj. R^2	0.736

	Est.	2.5%	97.5%	t val.	p
(Intercept)	7146971052.233	-47943906108.229	62237848212.695	0.255	0.799
Runtime	1556590.272	-139429164.202	142542344.746	0.022	0.983
imdbRating	95604819.941	38122635.160	153087004.723	3.265	0.001
imdbVotes	852.685	620.454	1084.916	7.208	0.000
tomatoReviews	34838.195	-608118.694	677795.085	0.106	0.915
tomatoFresh	792641.381	-1063986.763	2649269.525	0.838	0.402
tomatoUserRating	-5013114320.521	-37046181373.682	27019952732.639	-0.307	0.759
Budget	24.451	-4.880	53.781	1.636	0.102
imdbVotes_bin	-16665087.074	-50087116.717	16756942.569	-0.979	0.328
Budget_bin	120538438.376	10946511.701	230130365.051	2.159	0.031
Runtime_log	-1649155568.638	-16667247666.313	13368936529.037	-0.216	0.829
imdbRating_P2	-7929117.052	-13243645.500	-2614588.605	-2.929	0.004
imdbVotes_P2	-0.012	-0.039	0.014	-0.911	0.363
tomatoUserRating_P2	732253988.020	-3881824799.423	5346332775.463	0.312	0.755
Runtime:tomatoFresh	-6365.021	-13191.562	461.520	-1.830	0.068
Runtime:tomatoUserRating	-2434270.649	-83329526.068	78460984.770	-0.059	0.953
Runtime:Budget	0.069	0.004	0.134	2.082	0.038
Runtime:Budget_bin	-1233668.067	-2191096.141	-276239.992	-2.529	0.012
Runtime:imdbVotes_P2	-0.000	-0.000	0.000	-0.722	0.471
Runtime:tomatoUserRating_P2	432725.054	-11058069.835	11923519.942	0.074	0.941
imdbRating:tomatoFresh	64076.024	-222270.396	350422.444	0.439	0.661
imdbVotes:tomatoReviews	-1.015	-2.960	0.929	-1.025	0.306
imdbVotes:tomatoFresh	0.702	-1.450	2.855	0.640	0.522
imdbVotes:Budget	0.000	-0.000	0.000	1.924	0.055
tomatoReviews:imdbRating_P2	-39647.204	-57603.724	-21690.684	-4.334	0.000
tomatoReviews:tomatoUserRating_P2	110139.685	61417.807	158861.563	4.437	0.000
tomatoUserRating:Runtime_log	1144553360.845	-7561311693.714	9850418415.405	0.258	0.796
Budget:imdbVotes_bin	0.882	0.375	1.389	3.418	0.001
Budget:Runtime_log	-6.542	-14.307	1.223	-1.654	0.099
Runtime_log:imdbVotes_P2	0.003	-0.004	0.010	0.839	0.402
Runtime_log:tomatoUserRating_P2	-169217235.339	-1419272012.784	1080837542.106	-0.266	0.791

Standard errors: OLS